Universidade de Aveiro
Departamento de
Electrónica, Telecomunicações e Informática,
2018

Pedro Toipa Coelho

Um Registo Federado para Auto-soberania
A Federated Ledger for Regulated Self-Sovereignty

Universidade de Aveiro
Departamento de Electrónica, Telecomunicações e Informática,
2018

Pedro Toipa Coelho

Um Registo Federado para Auto-soberania
A Federated Ledger for Regulated Self-Sovereignty

""Code is law" in the sense that code will execute no matter what"

— Melanie Swan

Pedro Toipa Coelho

Um Registo Federado para Auto-soberania
A Federated Ledger for Regulated Self-Sovereignty

Dissertação apresentada à Universidade de Aveiro para cumprimento dos requesitos necessários à obtencão do grau de Mestre em Engenharia de Computadores e Telemática, realizada sob a orientação científica de André Ventura da Cruz Marnoto Zúquete, Professor Auxiliar do Departamento de Eletrónica, Telecomunicações e Informática da Universidade de Aveiro, e de Hélder José Rodrigues Gomes, Professor Adjunto da Escola Superior de Tecnologia e Gestão de Águeda da Universidade de Aveiro

o júri / the jury

presidente / president

Professor Doutor Joaquim Arnaldo Martins
Professor Catedrático da Universidade de Aveiro

Professor Doutor Hélder José Rodrigues Gomes
Professor Adjunto da Escola Superior de Tecnologia e Gestão de Águeda da Universidade de Aveiro

Professor Doutor António Alberto dos Santos Pinto
Professor Adjunto, Instituto Politécnico do Porto

agradecimentos / acknowledgements

I would like to thank my thesis advisers, André Zúquete and Hélder Gomes, for their insight, availability and overall support during this year of work.

I thank all my friends who helped build memories through this last year, I wouldn't be able to name them all.

Last but not the least, a very special thank to my family, in particular my mother, Rosa, for putting up with me during the ups and downs of my academic career, my sisters, Catarina e Raquel, for sharing their wisdom with me when requested, and when not requested, my brother-in-law, Bruno, for being the brother I never had, and my grandmother, Zalmira, for her great contribution in helping me become who I am.

Resumo

Num mundo onde as pessoas têm de se submeter a um alto escrutínio em cada processo que iniciam, e num mundo onde o ambiente digital cresce incessantemente, antecipa-se que cada vez mais serviços on-line peçam atributos pessoais e necessitem de confiar em tais atributos. Para lidar com essa necessidade, descrevemos uma proposta para um repositório seguro, descentralizado e compartilhado de atributos pessoais certificados. Os utilizadores beneficiam pois, têm controlo total sobre a divulgação de seu conjunto de atributos certificados (e.g., para comprovar a sua identidade junto dos fornecedores de serviços). As entidades certificadoras beneficiam de uma infraestrutura descentralizada persistente baseada em $blockchain$, evitando custos mais altos de uma infraestrutura centralizada sempre ativa, mantendo o poder de emitir e revogar atributos certificados. Finalmente, os fornecedores de serviços beneficiam da exatidão e dos atributos certificados que os indivíduos lhes revelam.

Abstract

In a world where people must subject themselves to high scrutiny in every process they initiate, and in a world where the digital environment grows incessantly, we anticipate more on-line services asking for personal attributes and their need to trust in such attributes. For tackling such need, we describe a proposal for a secure, decentralized and shared repository of certified personal attributes. Individuals benefit because they have full control over the disclosure of their set of certified attributes (e.g., to assert their identities to service providers). The certifying entities benefit from a resilient, decentralized infrastructure, avoiding the higher costs of an always-on centralized infrastructure, while retaining the power to issue and revoke certified attributes. Finally service providers benefit from the correctness and freshness of the certified attributes that individuals disclosed to them.

Contents

List of Figures

List of Tables

List of Acronyms

AA Attribute Authority

ACL Access Control Language

AP Attribute Provider

BFT Byzantine Fault Tolerance

BIP Bitcoin Improvement Proposal

CA Certification Authority

CLI Command Line Interface

Dapp Decentralised Application

DBFT Delegated Byzantine Fault Tolerance

DID Decentralised Identifier

DPoS Delegated Proof of Stake

DLT Distributed Ledger Technology

FBFT Federated Byzantine Fault Tolerance

GDPR General Data Protection Regulation

ID Identifier

IdP Identity Provider

IdM Identity Management

IoT Internet of Things

IPFS Interplanetary File System

JSON JavaScript Object Notation

MSP Membership Service Provider

PBFT Practical Byzantine Fault Tolerance

PIN Personal Identification Number

PGP Pretty Good Privacy

PoA Proof of Authority

PoS Proof of Stake

PoW Proof of Work

PKI Public Key Infrastructure

RA Registration Authority

RB Regulatory Body

SAML Security Assertion Markup Language

SP Service Provider

SSO Single Sign-On

VA Validation Authority

XML Extensible Markup Language

Chapter 1

Introduction

In 1993, Peter Steiner published a cartoon on the New York Times making a parable about anonymity on the internet [1]: "On the internet, nobody knows you're a dog".

Figure 1.1: Peter Steiner Cartoon about anonymity on the internet.

Twenty five years later, these words, presented on that comic, still, illustrate the challenge to identify individuals on-line.

Identity is a concept unique to the human being and it should not be confused with documents containing the owners Identifier (ID) (e.g., passport).

With the growing dependency from information technology, the concept of digital identity arouse and was linked to the identity register stored in databases [2]. Digital identity can be

defined by a set of attributes associated with the subject to whom the identity relates [3], which can themselves be defined as a piece of property or data about the person that can be attested by a trusted party [4].

As technology became a daily aspect of our lives, unconsciously, we started to create several partial identities for each service we may need, linked to an ID (e.g., an email address). This ID serves as an aggregator for a sub-set of attributes related to the respective service, and is generally protected by a password. And soon enough we started to see that a breach in one of these services would reveal personal attributes. In 2014, Sony Pictures was targeted by a group of hackers that leaked confidential data; on this leak there was also personal information of employees and their families, and salary information [5]. This example illustrates the risks of leaks of attributes managed by third parties.

Breaches like the one mentioned, could possibly lead to a snowball effect, were an attacker would be able to access other services sharing the same credentials. With events like this, we had to learn to segregate credentials to mitigate appearances in "*pwned*" lists. Unfortunately, humans are bad at remembering several different credentials, so cross-platform Identity Management (IdM) were deployed to help them. As Google, Facebook and other organizations, have accounts from a large number of users, they start providing an authentication service that attests the user identity to services from other companies, simplifying the user authentication by avoiding the need for the user to register in all those other services and to manage passwords for them all. We now have Identity Provider (IdP) that in exchange for that new authentication service they provide to relying services, gather more user attributes provided by those services, which is a relevant threat to the users' privacy.

Centralized models of IdM currently face challenges with the increase of data breaches that, ultimately, can lead to user privacy losses. This type of phenomena highlights the lack of control and ownership that end-users experiences with their digital identity. A decentralized service may fill some of the flaws present in a centralized service, in a decentralized scenario the identity information is referenced in a ledger that no single central authority owns or control. Fortunately, Distributed Ledger Technology (DLT), is today a mature decentralized solution with several possible uses; one of them is identity management and user control of its identity [6]. The scenario where the user is in complete control of their identity, how it is stored, how and to whom it is revealed, is now possible, and has recently begun to be associated with the term: self-sovereign identity.

The work presented in this document will be focused on introducing self-sovereign identity applied to personal attributes.

1.1 Motivation and Problem

When using Google or Facebook as a way to login in other relying services, a user, inevitably, agrees to share with these services some information about themselves, i.e., attributes from their identity. Some services only need these attributes and rely on the IdP as a mean to guarantee that those attributes are somewhat true. Sync.me[1] is a prime example of a service that uses Google or Facebook to gather information about the user, the goal of the service is to build a global phone book, for this, it uses the list of contacts that the user saves in the aforementioned providers. The type of interaction mentioned above can be summarized in

[1] https://sync.me/

two aspects, a party has interest in accessing an attribute and the user has interest in sharing that attribute.

This can be particularly interesting with attributes that need to be certified by a trusted entity, such as an academic certificate. This document serves as a proof of a competence acquired by the subject, i.e., it is an attribute of the subject. In a traditional scenario, the subject would have a printed copy with an official stamp and hand-written signature that can be presented to interested parties. In alternative, a copy can be made but would have to be signed by a notary to prove its origin.

Even though organizations tend to provide attributes digitally, through a proprietary platform or other means, in many cases an attribute will still be linked to a physical document.

With the amount of web services available to us nowadays, it is very easy to encounter limitations because a service may need to verify an attribute presented in a physical document. This led to the research presented in this paper and to the formulation of the following problem: *How can we certify attributes in a connected world without prejudice to the subjects' privacy and giving them control over their personal attributes.*

1.2 Contribution

This dissertation describes a solution to give an individual complete control over disclosure of a specific set of their identity attributes, those that are certified by a trusted third party. For this we must distance ourself from centralized entities that own attributes that belong to users, and move towards the edge were the user stands.

Having in mind the removal of control from a central entity, we are looking at a decentralized solution. As this is the core feature of Blockchain, and as DLTs main goal is to provide decentralized trustless networks, it can be conveniently mapped to fit our goals.

Thus we propose a system to manage attributes while conforming with self-sovereign identity. In addition it must also comply with the following requirements:

- **Data Security**: Identities and their respective attributes must be stored in a secure fashion, possibly protected by cryptographic functions;
- **Issuer Validation**: An issuer must only be able to certify attributes under its jurisdiction;
- **Unlinkability**: Knowing one or more attributes from a user must not give the opportunity to know all the remaining attributes from that user. Entities with access to an user attribute must not know which other entities can access it and for what purposes;
- **Revocation**: Attributes must be revocable by the respective issuer entity;
- **Repudiation**: It must be possible for users to repudiate attributes issued on their behalf;
- **Usability**: To interact with the system, the necessity to understand the underlying technologies, such as blockchain, key management procedures or encryption protocols, should not exist.

During the research process, from which this dissertation resulted, a project proposal was submitted, in December 2017, to *Imprensa Nacional-Casa da Moeda* under the *Prémio Inovação INCM 2017* grant. Unfortunately the project was not selected.

The research also led to the publication of a paper entitled "A propose for a federated ledger for regulated self-sovereignty" in CISTI'2018 - 13th Iberian Conference on Information Systems and Technologies [7]. The paper aimed to present the idea of a regulated ledger for self-sovereignty and discuss some of the necessary requirements and possible technology solutions to solve them.

As a result of the publication in the CISTI'18 proceedings, we were invited to publish in the Journal of Information Systems Engineering & Management, the opportunity was taken to update the idea with the work developed since then. The article, entitle: "Federation of Attribute Providers for User Self-Sovereign Identity", is currently submitted for review.

Chapter 2

Digital Identity

2.1 Introduction

2.1.1 Definitions

A digital identity is a representation of an identity that takes part in an on-line transaction. It is unique and usually confined to a service or context. This does not mean that the real life identity must be known, as this uniqueness does not have to be transversal between contexts [8].

There are several components linked to the digital identity concept, and to better understand the platforms that deal with digital identity we need define some of the core concepts.

Identifier

An identifier can be seen as the answer to the question "Who are you". It is used to distinguish different entities in a system. A entity can have different identifiers, this can be dictated by the function or context. Examples of identifiers are: the name, Citizen Card number, Social Security number, student number, among many others. Depending on the needs of the specific service an user may want to obtain, different identifiers may be used [9].

Authentication

Authentication of an entity occurs when proof of correlation between user and identity is provided. Generally the system provides a challenge on which the user can apply a transformation based on some secret or private information (token) or provide the system with a secret, the token and secret are known as authenticator. In the later, authenticators can be passwords, Personal Identification Number (PIN) or private keys. For passwords and PINs the system will perform a comparison with stored values in its possession. With private keys the process is different, as no secret (private key) is transmitted and the system uses the user public key to verify a proof (the result of a transformation) that the user is in possession of the corresponding private key.

The classic paradigm for authentication identifies three factors as cornerstones [8]:

1. **Something you know** (e.g., a password)
2. **Something you have** (e.g., an ID badge or cryptographic key)

3. **Something you are** (e.g., biometric data)

The combination of more than one of these factors leads to Multi-factor authentication

Authorisation

Authorisation is the function of defining rights or privileges to something or someone to access some resource or perform some action. Access control can then evaluate if the rights given are sufficient in order to access the intended resources [10].

Attribute

Attributes are pieces of data that form an identity. These values can be persistent, ephemeral or mutable (i.e. their value may vary over time). As examples of attribute we can consider age, employer or address.

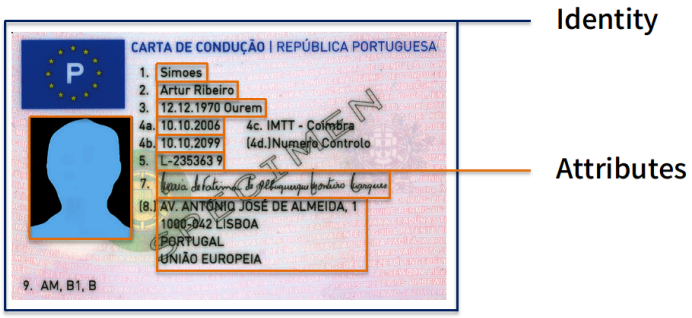

Figure 2.1: Drivers License composed by several personal attributes

Figure 2.1 shows a drivers license (representing the user identity) composed by the the attributes described by the values in the respective field.

Attribute Attestation

Attestations or claims are assertions that maps a certain attribute to an identity. In some identity management systems, attribute verification follows a successful authentication, this confirms that the identity presented is the correct one. An attribute attestation can be a bank attesting to a certain individual having a determined bank account with a certain number.

Federation

A Federation is a collection of domains that establish trust and that facilitates authentication and subscriber attribute verification across networked systems. In a federation the verifier is referred as an IdP [11].

2.1.2 Self-Sovereign Identity

Self-Sovereign identity is a user centric approach to identity. It mandates that the individual should have full control and autonomy over their identity and data.

The cryptographer Christopher Allen defines ten principles for self-sovereign identity[1]:

1. **Existence**: *Users must be independent from the system.* An identity, particularly a self-sovereign one, is based on the "I" and can never be completely digital, so in the context of self-sovereign identity only some limited aspects of the "I" are made public and accessible.

2. **Control**: *Users must control their identities.* The users are the maximum authority on their identity. They decide how and to whom their identity is disclosed and used. They should always be able to refer to it, update it or hide it. This does not mean a user controls all claims on their identity, other entities can make claims about a user.

3. **Access**: *Users must have access to their own data.* Users own identity should always be readably available to them. Without meaning that they can freely alter their claim, but it means they should always be aware of them.

4. **Transparency**: *Systems and algorithms must be transparent.* The processes and system used to manage data, and the algorithms used to analyse it must be well-known, open-source and preferably independent of any particular architecture.

5. **Persistence**: *Identities must be long-lived.* Identity should last for ever or as long as the user desires. This must not contradict the "right to be forgotten".

6. **Portability**: *Information and services about identity must be transportable.* Identities should not be owned by third parties, even if trusted. Allowing an identity to be transportable means that users remains in control of their identity even in a change of regime or jurisdiction.

7. **Interoperability**: *Identities should be as widely usable as possible.* Identities must operate across systems, companies and borders. Their value diminishes when their range of usage is limited to niches.

8. **Consent**: *Users must agree to the use of their identity.* Sharing of data must occur only with the consent of the owner of said data. Even if a party offers a claim about a user, the user must consent to its validity, this does not have the need to be interactive but should still be deliberate.

9. **Minimalization**: *Disclosure of claims must be minimized.* When data is disclosed, this discloser should involve the minimal amount of data possible. As an example, if only a minimum age is asked, then the exact age should not be disclosed, and if only an age is requested, then the more precise date of birth should not be disclosed.

10. **Protection**: *The rights of users must be protected.* When a conflict between the needs of the system and the rights of the user arises, the system should fail in order to preserver the rights of the users.

These may seem very strict principles, but all are required in a truly self-sovereign system. The General Data Protection Regulation (GDPR)[2], introduced in Europe this year legislates some of this principles and increases the importance of research in this context.

[1] https://www.lifewithalacrity.com/2016/04/the-path-to-self-soverereign-identity.html
[2] https://www.eugdpr.org/

2.2 Identity Management Systems

The enterprise environment brew the need to unify authentication and/or authorization systems in order to achieve easier management and better security. This need would rapidly evolve to full fledge systems capable of dealing with the problem of fragmented multi-account, referred in the first chapter, by centralizing the user identity data and standardizing its usage.

Some approaches are summarised bellow.

2.2.1 OAuth

OAuth [12, 13] is an open standard for authorization. OAuth enables an application, the client, to take action and access resources on a resource server on behalf of the user, without the need to share credentials with the client.

The goals of OAuth are achieved by allowing the issuing of tokens by an IdP to the third party applications. This token is then used by the client to access said resources.

Although OAuth is dedicated to authorization, it can be used as a form of pseudo-authentication. This pseudo-authentication relies on a twist of the authorization process. The service will suppose that if the token provided by the user accesses resources from the user, then it means that the user is indeed who he claims.

2.2.2 OpenID

OpenID is an open standard for authentication[3]. With OpenID an user must choose and obtain an OpenID account through an IdP supporting the standard. The user shall then use this account to login in into third party services.

OpenID Connect

The current version of OpenID is OpenID Connect[4]. This is an identity layer built on top of OAuth 2.0, allowing clients to verify the identity of the user based on the authentication performed by the authorization server. On top of this, it also allows the client to access several basic information about the user using REST-like services. There are proprietary standards that operate in a similar fashion, e.g., Facebook Login[5].

2.2.3 SAML

Security Assertion Markup Language (SAML) is a framework based in Extensible Markup Language (XML) for communicating user authentication, entitlement, and attribute information[6].

SAML enables web Single Sign-On (SSO) through the communication of an assertion from one service to another. This second service, if confident of the origin of the assertion, can then accept the user authentication as if it was done directly.

An other interesting use case of SAML is attribute-based authorization. Where a party informs other party on access attributes that the user possesses. In this case, this information

[3] https://openid.net/
[4] https://openid.net/connect/
[5] https://developers.facebook.com/docs/facebook-login/
[6] http://saml.xml.org/

can be limited to some attributes of their identity instead of the whole set, or as supplement, how and when this user was authenticated.

2.3 Public Key Infrastructure

A Public Key Infrastructure (PKI) [14, 15] has the intent to manage public key encryption and certificates. It also facilitates authentication of parties in a network and secure transfer of information. It is composed by set of roles, policies and procedures to create, manage, use, store, and revoke digital certificates.

A PKI binds entities with their respective public key. The binding is established through the registration of the entities, by a Registration Authority (RA), followed by the issuance of certificates by a Certification Authority (CA). This process may depend on the level of security required. Different levels of security may apply to this process depending on the requirements.

In public key cryptography, each entity owns a key pair, composed by a public and private key. These two keys have a relation such that data encrypted with one key can only be decrypted with the other key. The public key of an entity is well known and widely distributed, while the private key must be kept secret and safe by the owner. This private key can be used to authenticate the user, i.e., to prove that they are the owner of a public key in a certificate and, consequently, that they are the owner of the identity attributes contained in the certificate, typically a name.

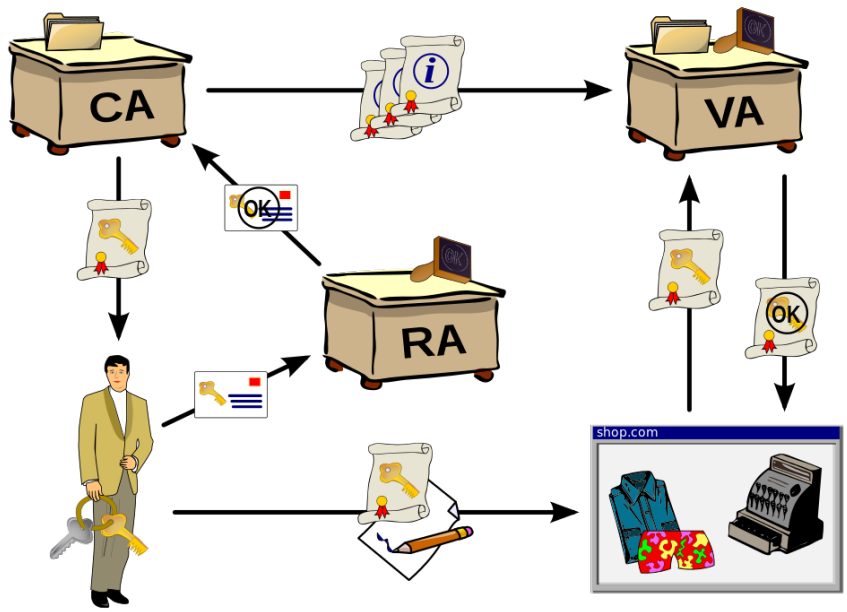

Figure 2.2: Public Key Infrastructure Diagram. A user applies for a certificate with his public key at a RA. The latter confirms the identity of the user to the CA which in turn issues the certificate. The user can then digitally sign data using his new certificate. His identity is then checked by the interested party with a VA which again receives information about issued certificates by the CA[8].

2.3.1 Digital Certificates

Strong public key, as read only objects, provide some protection against forged keys. However, an evil doer could pass as the secure directory containing this key, allowing him, instead of the intended recipient, to decrypt messages. Thus, we need a way to validate public keys. Digital certificates provide key validation.

A certificate is a digital document that binds a public key to an entity (a person, application, service, etc.). These certificates contain information about the issuing entity, the subject of the certificate, expiration date and a digital signature. This signature ensures that the issuing entity is verified [16].

2.3.2 Certification Authorities

A Certification Authority (CA) issues certificates, which are signed documents containing a public key linked to some identity attributes (typically, the name) from a given entity. For this, it needs to act as a trust anchor for both the subject entity of the certificate, the certificate owner, and the entities relying on it.

In other words, a CA is a centralized entity that provides individual validation for every entity that wishes to be certified.

It was already establish that an entity is verified by a CA, this CA can also be verified by other CA, and so on until it reaches a root Authority. This is a particular type of CA that is inherently trusted by the user and/or system.

2.3.3 Attribute Authorities

An Attribute Authority (AA) [17] works in a similar fashion to a Certificate Authority. When there is the need to separate public keys and attributes, an AA can be used to publish an attribute certificate.

The use of Attribute Certificates may be useful when the related public key certificate life-time possesses a different life-time from the one of the attribute, or vice versa. It is also useful when a CA does not have authority over an attribute.

2.3.4 Signatures

Digital signature act as proof of authenticity and integrity of a piece of data. It guarantees that digital signed information was produces by a trusted source and that its content remained the same since the signature.

Data goes through an hashing functions that produces an output uneasy to reverse engineer. This hash is then encrypted with the private key of the entity and sent with the data. The receiver can validate the message integrity by using the same hashing functions and matching it with the decrypted signature.

2.3.5 Web Of Trust

With the introduction of Pretty Good Privacy (PGP) 2.0 it was also introduced the concept of web of trust. Web of trust refers to a decentralised model of key validation achieved via peer validation of public keys. The process occurs when a user identifies public keys as belonging

[8]Image source: https://en.wikipedia.org/wiki/Public_key_infrastructure

to a certain individual, a chain of trust can then be followed until a trusted user is reached [18].

The difference between a typical PKI and PGP resides that in the former the certificate is issued by a CA, which are organized in a hierarchical structure, and in the later any entity can issue a certificate. When validating, in the hierarchical model trusting the root is enough to validate any certificate, however, in the distributed model, the validation implies that we trust the entity that issued it.

Chapter 3

Blockchain

3.1 Introduction

A blockchain is a continuous growing list, or ledger, of records. In this type of ledger the records are called blocks and are linked and secured with cryptographic functions. This linkage is implemented by including in each block a cryptographic hash of the previous block, blocks include a timestamp and a list of transactions processed.

In 2008, Satoshi Nakamoto [19] introduced a blockchain as the cryptocurrency Bitcoin. Bitcoin was the first digital currency to address and solve the problem of double spending without the need of a centralized party.

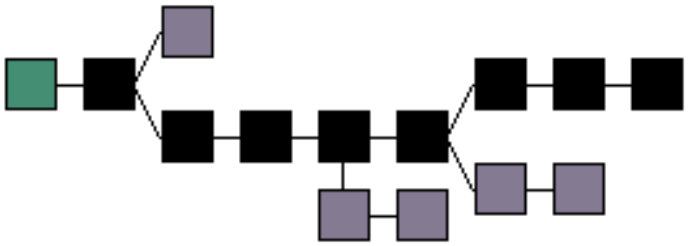

Figure 3.1: Block sequence on a blockchain[1].

The design of a blockchain offers, the data in it, a natural immutability and persistence. Such immutability is achieved due to each block containing the hash of the previous block, any alteration to a block would alter the hash, this would imply a change in all subsequent blocks in order to keep the chain valid, and this is extremely hard, or impossible, because of the consensus mechanism employed. Once added, a piece of data cannot be retroactively altered. This properties makes this type of ledger a desirable mean to maintain a permanent and verifiable record of transactions between parties.

A blockchain will run in a peer-to-peer fashion where each node of the network contains an integral copy of the ledger. This requires that all the nodes agree to the information being stored and that all of them have the truest and most recent chain to consult or work on. The need for a consensus between all the nodes can be resolved with many different decentralized consensus strategies.

[1]Image source: https://en.bitcoin.it/wiki/Blockchain

3.2 Digital Wallets

In networks dedicated to cryptocurrency a digital wallet exists to hold the currency. A party that owns such a wallet has a set of public and private keys. The wallet is used to store the set of asymmetrical keys used in operations related to the ledger.

A digital wallet is sometimes referred as the coins address, where this address is one of the public keys in the wallet that is used to identify the receiver of a transaction, the private key of the pair is used to authorize transactions and prove ownership.

In summary, the public key is an alias of identity on the network as it cannot be directly linked to the individual that owns it.

3.3 Hierarchical Deterministic wallets

A deterministic wallet is a simple method of generating an address from a seed, which the User owns and controls. Bitcoin Improvement Proposal (BIP) 0032[2] describes a type of deterministic wallet that can generate a near infinite number child keys from the seed key.

The child can operate independently from the parent and the parent will continue to be able to operate even if the child key is compromised, as the relation bottom-up is invisible.

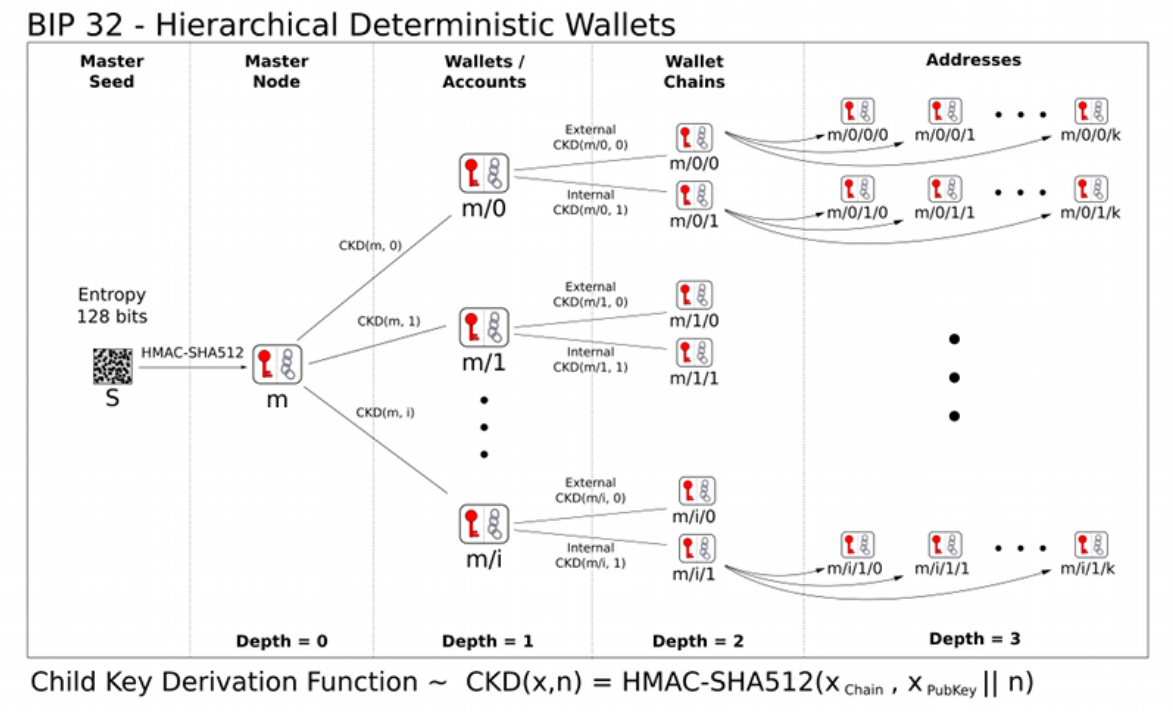

Figure 3.2: Hierarchical Deterministic wallets: given a seed key and a index, a cryptographic function derives several child keys[4].

[2]https://github.com/bitcoin/bips/blob/master/bip-0032.mediawiki

[4]Image source: https://github.com/bitcoin/bips/blob/master/bip-0032.mediawiki

Given that the public key is an alias of identity of the user, this mechanism facilitates the management of multiple keys from the same user, this can be interesting in a scenario where we have interest in improving an user privacy.

3.4 Transactions

A transaction represents a transfer of tokens, where a token is some asset, like currency in Bitcoin. In Bitcoin, transactions must include an identification of the recipient (a public key), and must be signed (using a private key) by the sender. These transactions will then be put in a block that must be inserted into the ledger. The process related to block insertion in the ledger is related to consensus mechanism that will be address in the next section.

Figure 3.3 illustrates the flow of a transaction in the Bitcoin network. Alice intends to send Bob some digital currency, for this Bob has to give his wallet address to Alice, only then can she create a new transaction. This transaction contains the address from Bob account, the amount of coin, and the fee needed to finance the consensus. Alice will sign the transaction with her private key, this will give the network a cryptographic proof that Alice owns the origin address and can operate those coins. After this step the transaction is propagated through the network, where it is received by other machines, known as miners, that will lend their computing power to build the blocks.

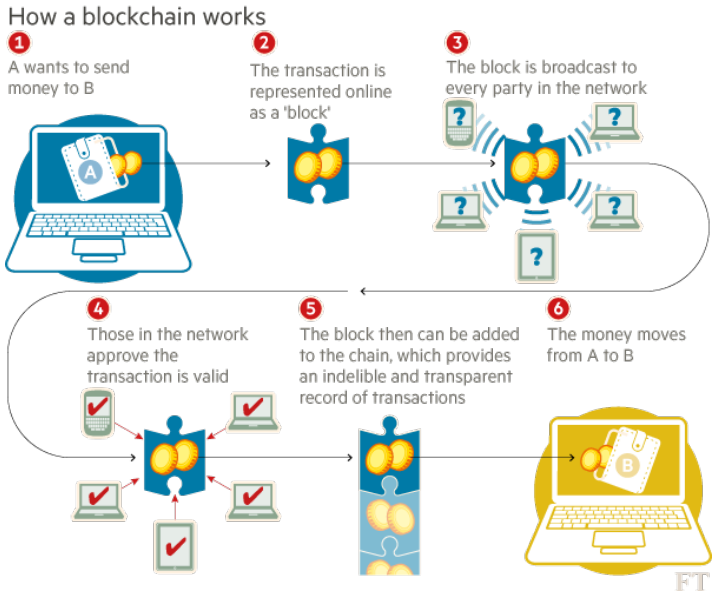

Figure 3.3: A transaction flow through the Bitcoin network[5].

3.5 Consensus

Using a blockchain to replace a traditional centralized ledger with a distributed one creates the need to ensure that all the transactions are correctly ordered and synchronised.

[5]Image source: https://halpernfinancial.com/views/what-is-blockchain-and-why-should-i-care

Given that any party can submit new information to the blockchain, it is necessary for the nodes to evaluate and agree on this information before it can be permanently added to the chain.

In the Bitcoin network, where we deal with a currency, the new information may be related with payment of services, the new addition of data will also imply a verification to validate if there is enough currency to perform the transaction. Alice sends a transaction with a certain amount of currency to Bob, the network needs to ensure that Alice owns that amount of coin and that is not being spent twice. An evil doer may try to submit the same coin in very close instant of time, this is called double spending, and represents a problem intended to be solved by consensus algorithms [20].

Figure 3.4 illustrates this problem and how the application of a consensus algorithm may help counter it.

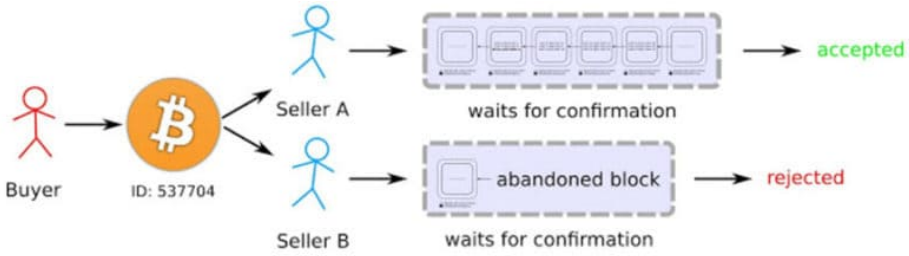

Figure 3.4: The Double spending problem on Bitcoin is solved by choosing the longest chain calculated by the peers [7].

3.5.1 Proof of Work

Bitcoin employs Proof of Work (PoW) [19] as consensus algorithm. This algorithm relies on nodes competing on hashing power to allocate accounting rights and rewards. Based on the information of the previous block, the nodes need to calculate a specific solution for a complex mathematical problem. The first node to solve this problem is granted the ability to create the next block, and will be reward with a certain amount of currency.

PoW uses workload as safeguard. All nodes trust the longest chain and the length of the chain is proportional to the amount of workload. For one to control the blockchain they need to control more than 50% of the hashing power and ensure that they can create the latest block.

3.5.2 Proof of Stake

Proof of Stake (PoS) [21] is the most common alternative to PoW. PoS depends on participation, it assumes that the more participation a node has in the network the less interest it has to commit fraud, as this is will only harm their interests. There are several measures to determine participations, such as amount of currency, age of currency, and others.

PoS tries to solve PoW problem of wasted resources. The security of the blockchain is proportional to the value the miners have invested in the network. To devise an attack, the

[7]Image source: https://coinsutra.com/bitcoin-double-spending

attacker would have to accumulate a large number of coins and hold them long enough to make the attack.

Delegated Proof of Stake

Contrary to PoW and PoS, in Delegated Proof of Stake (DPoS) [22], the miners work together to make blocks instead of competing. In DPoS, coin holders do not vote on the validity of the blocks, instead they elect witnesses or delegates to make this validation.

3.5.3 Proof of Authority

Proof of Authority (PoA) [23] ensures validation of transactions resorting to approved accounts. These accounts are the authority from who other nodes receive their truth from. Although PoA has a high output, it deposits the decision process on a single node, this creates a central point of failure, and it can even be said that this approximates the network to a centralized one. This type of consensus is ideal for a private network.

3.5.4 Byzantine Fault Tolerance

The Byzantine Generals problem explains a classic problem in distributed computation. The problem refers that several generals and their armies have laid siege to a city. The decision to invade must be unanimous. If some generals give the order to attack without the others the battle will be lost. This means that the generals will have to use messengers to communicate with one another in order to better plan their moves [24].

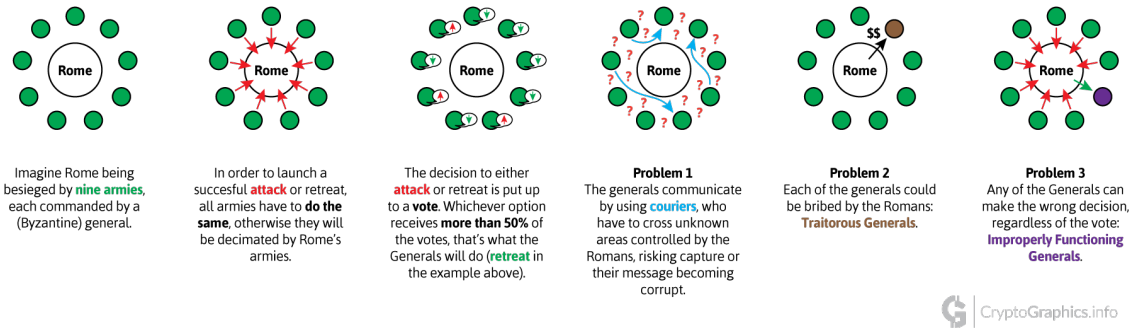

Figure 3.5: The Byzantine Generals Problem[8].

Several blockchains adopt algorithms that employ some versions of Byzantine Fault Tolerance (BFT).

Practical Byzantine Fault Tolerance

Practical Byzantine Fault Tolerance (PBFT)[25] relies on the sheer number of nodes instead of hashing power in order to confirm transaction and generate trust.

[8]Image source: https://cryptographics.info/cryptographics/blockchain/consensus-mechanisms/delegated-byzantine-fault-tolerance-dbft/

In PBFT every node as a public key published. Every message passing through a node must be signed by the node to verify its format. Once enough identical responses are reached, then you can agree that the transaction is valid.

Federated Byzantine Fault Tolerance

The Federated Byzantine Fault Tolerance (FBFT) consensus mechanism are generally associated with Stellar [26] and Ripple Blockchains.

In this type of consensus the nodes do not have to be known and verified ahead of time. Membership is open, and control is decentralized. Nodes choose their pool of trustees.

Consensus is reached when a system wide quorum emerge from individual nodes, but a part, or slice, of this quorum can be enough to convince an individual node and sway is vote.

Delegated Byzantine Fault Tolerance

In Delegated Byzantine Fault Tolerance (DBFT) consensus occurs through gamification of block verification among professional nodes. These professional nodes are the ones participating in the network that are trying to make a profit. The professional nodes participating in the consensus are nominated by all other nodes. After a professional node propagates their version of the network, other 66% of the nodes need agree on it in order to achieve consensus. If this criteria are not met another node must be nominated to broadcast their version until consensus is met[9].

3.6 Smart Contracts

The term smart contract was first conceived by Nick Szabo [27] as "a computerized transaction protocol that executes the terms of a contract". In other words contractual clauses should be translated into code and embedded into software or hardware, the goal was to minimize the need for trusted intermediaries between the interested parties, and the occurrence of malice and undesired exceptions.

The recent surge of DLT brought the concept back to life, and its coined as one of the main features on the Ethereum network. In this context, a smart contract is a script stored within the ledger. As they are stored in the chain they possess an address, ordering a transaction to this address will trigger its execution on every node of the network.

A smart contract functions as an autonomous actor, whose behaviour is predictable and deterministic, meaning that given the same input it will always produce the same output. A smart contract can be seen as having its own "account" on the blockchain, he can also muster assets and manage them as its code mandates. It will also sign every operation that it does, leaving a trace that can be verified by the other participants. Its code is also verifiable given that it resides in the network, which makes it public.

Developers can also create smart contracts that provide features to other smart contracts, similar to how software libraries work. Or smart contracts could simply be used as a way to store information.

[9] https://cryptographics.info/cryptographics/blockchain/consensus-mechanisms/delegated-byzantine-fault-tolerance-dbft/

The properties of a smart contract means that it can be trusted to execute any on-chain logic, as long as this logic can be expressed as functions of data inputs, and as long as the smart contract is able to reach the data that it was intended to manage.

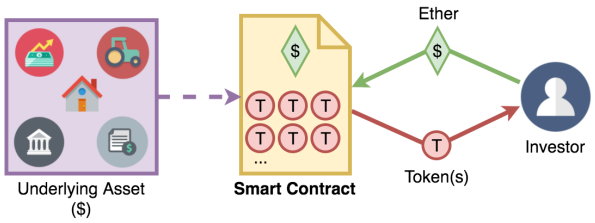

Figure 3.6: Smart Contract managing resources. A smart contract can represent a physical asset (e.g. stocks), this smart contract offers an investor the possibility to buy a stake of the asset[11].

3.7 Types of Blockchain

A blockchain can be categorized by the openness of the network: Public, private and permissioned[12].

	Public	**Private**	**Permissioned**
Topology	Decentralized	Partially decentralized	Partially decentralized
R/W access	Anyone can read and anyone can write	Read and Write permissions are controlled by single authority, i.e., the owner	Read and Write permissions are controlled by a predetermined set of nodes
Advantages	Secure, as the entire network verifies the transactions; Transparent as all transactions are made public	Efficient, as verification is made by only one node; Control of Read and Write permissions	Efficient, as there is a small number of nodes participating in the verification; Control of Read and Write permissions; No consolidation of controlling power
Disadvantages	Inefficient	Control resides in only one organization; May need some sort of node authentication	Needs some sort of node authentication

Table 3.1: Summary of blockchain technologies categorization

[11]Image source: https://docs.web3j.io/smart_contracts
[12]https://blockchainhub.net/blockchains-and-distributed-ledger-technologies-in-general/

Generally, efficiency and performance grow inverse to the decentralization and openness of the network. The reason behind this is bigger amount of resources and time needed to reach consensus.

3.7.1 Public Blockchain

A public blockchain is the classification given to a ledger where anyone can read or send transactions freely.

In a public blockchain every node shares equal right to read and write data in the ledger. Anyone can participate in the blockchain and in its consensus algorithm. The absence of an central entity dictating rules means that this type of network can be totally decentralized.

A public blockchain fits a use case where total transparency is specially needed and trustless to third parties. Bitcoin is a good example of this type of implementations as it was also the pioneer of it.

3.7.2 Private Blockchain

A private blockchain implies a fully centralized structure. Having the ability to restrict permissions to access data increases the privacy of the participants in the network when compared to a public solution. However the central entity who controls the network has full privileges and power to make decisions and change rules as it pleases.

This type of implementation suits a scenario where public readability is expendable.

3.7.3 Permissioned Blockchain

Permissioned blockchains, also called consortium or federated blockchains, represent a compromise between the two previously solutions. It maintains a certain level of decentralization while offering the possibility to control read and write rights. This rights depend on the identity of the participant and the role it plays as stakeholder. This type of networks usually depend on smart contracts to enforce the logic and identity validation before executing transactions.

Hyperledger is an example of implementation of a permissioned blockchain. It is a open-source distributed ledger designed to record and share data as well as maintaining privacy and security.

Chapter 4

State of the art

The need to answer to the question "Who are you?" about an entity behind a device connected to a network it is not a new problem. In the seventies a public directory was envisioned by the inventor of public-key cryptography, and a decade after, a grand scheme of hierarchical certification was envisioned. But the solution that stands is the one designated by a "patchwork of identity one-offs" [28], comprising several identity management systems that are restricted to specific domains without interaction between them.

With the rising of DLT, from which blockchains stand out, the opportunity to develop technological infrastructures, capable of providing the individual ownership and control over their personal data, grew accordingly. Having this possibility, for the first time in history, we witness arise of many proposals towards self-sovereignty identity.

In this chapter we are going to take a look and outline what as been already achieved in the path of self-sovereignty identity, particularly when it includes validations of personal attributes.

4.1 Sovrin

Sovrin [29] uses an open-source DLT, Hyperledger Indy, to provide an identity network. In Sovrin only trusted institutions can run nodes that participate in the consensus protocol (stewards). Making this particularly DLT into a permissioned one.

The governance of the network rest on the Sovrin Foundation and their a legal agreement called the Sovrin Trust Framework.

A user can use Sovrin to manage any kind of attribute, certified or not.

Within Sovrin a user can generate as many identifiers as they wish, to keep a separation of identities in order to protect their privacy, these identifiers are unlinkable and controlled with different asymmetric keys. This project uses the Decentralised Identifier (DID) specification structures containing the user identifier, a cryptographic public key and other relevant metadata[1].

Sovrin ledger acts a root-of-trust, and, as new organizations join the network they also can become trust anchors (i.e., add new organization and new users).

A mobile application and control software agents provide interaction with Sovrin. These agents allow the user to connect with Service Provider (SP) agents. These agents are endpoints that are always addressable and accessible. They may run in server owned by the users, but

[1] https://w3c-ccg.github.io/did-spec/

the most likely scenario are that specialized intermediaries will be used. This intermediaries, or agencies, alike e-mail systems, will run the agent for the user. Agents also provide backup service and encrypted storage of credentials.

This application also manages the cryptographic keys stored in the mobile device of the user. In case of theft or loss a key recovery mechanism is offered. This relies on a set of trustees. In the event of a recovery request, a quorum of trustees must sign a new identity record transaction that will then be verified by the stewards.

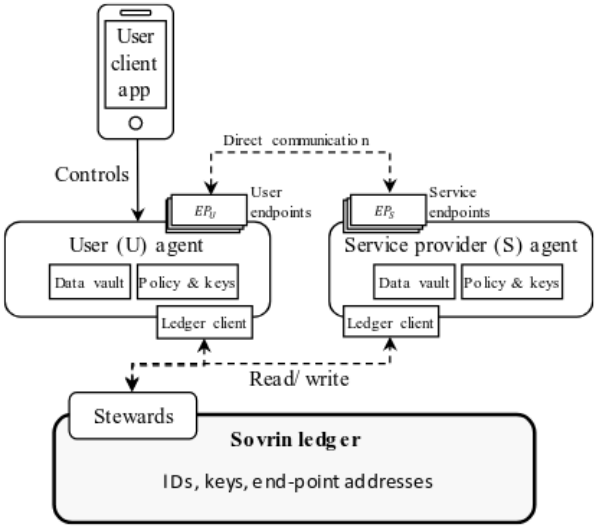

Figure 4.1: At the base of Sovrin is a permissioned ledger. Only stewards that legally abide by the Sovrin Trust Framework can write to the ledger. Users and organisations rely on agents that are addressable network points. Identifiers, keys and endpoint address are stored in the ledger.

4.2 uPort

uPort [30] aims to provide a decentralized identity framework. Its use case is linked to identity management in Ethereum Decentralised Application (Dapp)s and traditional centralised applications.

A uPort identity is supported by the relations that Ethereum smart contracts can establish with one another. uPort has two templates of smart contract: controller and proxy. When creating a new identity the user will generate a new key pair on their mobile device through the uPort mobile application. Then it will send a transaction which instantiates a controller containing the reference to the public key. After this, a new proxy with the controller reference is created, only the controller can call functions from the proxy. A user is free to create multiple uPortIDs that are unlinkable.

The user private key is stored on their mobile device. This leads to an important aspect of uPort. Like Sovrin, there is the possibility to recover a from a lost key. The user must nominate trustees that can vote to replace the public key of the user. When consensus is achieved between the trustees the controllers replace the lost key with the proposed public

key. This allows the user to maintain a persistent uPortID even after a loss of keys.

uPort also maps identity attributes to a particular uPortID. This is done by the registry, a global mapping of IDs to attributes. Any entity can query a registry but only the owner can modify it. Given that it is inefficiency to store large volumes of data on the ledger, only the hash of the JavaScript Object Notation (JSON) attribute structure, is stored, the data itself is stored on the Interplanetary File System (IPFS), a distributed file system where a file is mapped by its cryptographic hash.

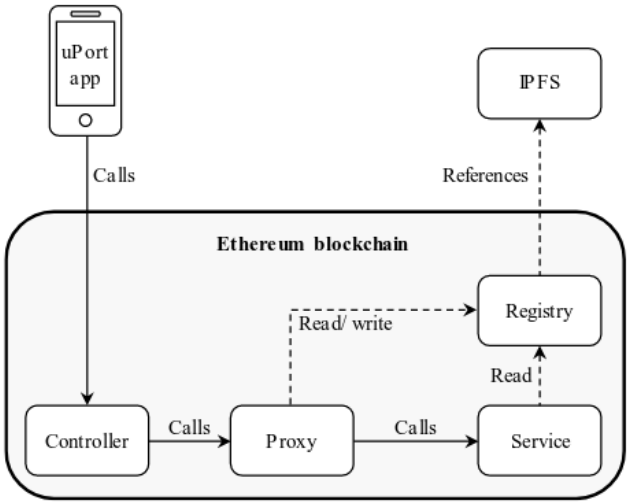

Figure 4.2: On uPort a user identity is comprised of a mobile application and two smart contracts: controller and proxy. The registry is a smart contract that provides a decentralised mapping of uPort identifiers to identity attributes. The registry can be globally read, and can reference data stored in an off-DLT data store such as the IPFS.

4.3 Blockcerts

Blockcerts[2] is a initiative from the Media Lab of the MIT and Learning Machine. The goal of the project is to issue and validate official records such as academic achievements. It uses version 2 of the Open Badge standard for its certificates expanding the former with distributed ledger technologies.

In Blockcerts, a certificate issuer signs a well-structured digital certificate and stores its hash within a blockchain transaction. The output is then assigned to the recipient of the certificate. Besides this linkage to the public key of the owner of the certificate, this one will also contain the email of the owner to enforce the linkage of identity. When presented to a third party, this party can attest if the certificate actually belongs to that user through a verification email and/or with a challenge-response which the user can use to prove that he possesses the private key that pairs with the known public key.

The key pair should have been previously generated by the user and stored in a mobile application built for the propose of storing this key material and the collection of certificates.

[2]https://www.blockcerts.org

Blockcerts addition to verify the validity of a certificate resides in the blockchain and the hash calculated at the moment of issue. The integrity is checked by looking at the blockchain and see if the hash is indeed there and if it corresponds to the original

In the first version of the Blockcerts a small amount of cryptocurrency would be assigned both to the issuer and the owner. If this amount where to be spent the verifier should consider that the certificate as been revoked. In later versions a unique revocation list could be assigned to each certificate.

The usage of this revocation list, as well the standard libraries to verify the validity of the certificates, allows the issuer to know how often a certificate is shared and possibly with whom is shared.

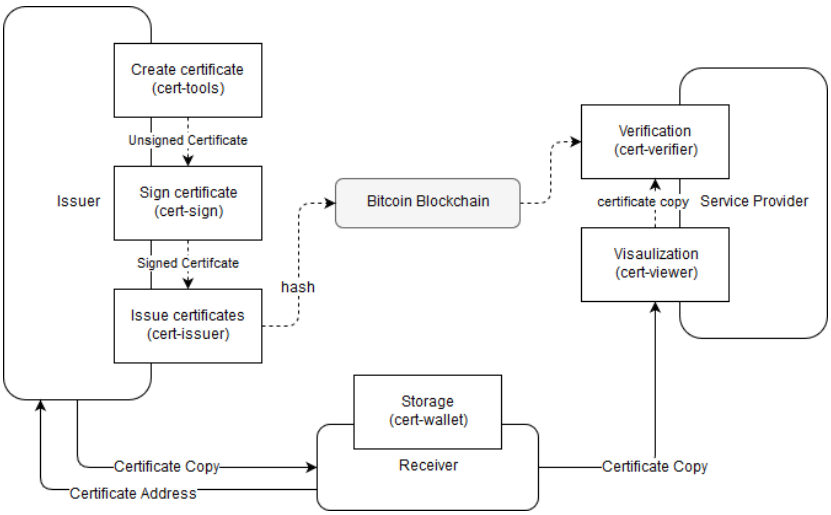

Figure 4.3: The Blockcerts architecture has two phases, issuing, on the left side, and distribution, on the right side. The issuance depended on an address given by the receiver. The certificate is created, signed, hashed to the blockchain and a copy given to the receiver. The receiver will then distribute this copy and the service provider will verify its hash on the address that corresponds to the certificate.

4.4 ShoCard

ShoCard [31, 32] provides a trusted identity that recurs to DLT to bind a user ID to an existing trusted credential (e.g., personal identification document), with additional personal attributes. This is made through cryptographic hash stored in Bitcoin transactions.

ShoCard uses a central server as a core element of its architecture, this server intermediates the exchange of encrypted identity information between the user and relying parties. This architecture is dived in three parts: bootstrapping, certification and validation.

Bootstrapping occurs when a new ShoCard is created. The mobile app generates cryptographic material for the user and scans their personal identification document through the mobile device camera. The image obtained and respective data are then encrypted and stored in the device, the signed hash of this data is then embedded into a Bitcoin transaction for later validation purposes. This transaction will be known as the user ShoCardID and is stored in the app as a pointer do the ShoCard seal.

The interaction with IdPs to gather attributes is made in the certification processes. During this process the IdPs must ensure that the user knows the data hashed and the keys that signed the seal. Face-to-face validation can be achieved by the user providing the original data used to produce the seal, a digitally signed challenge and the original trusted credential. The certificate takes the form of a signed hash of new attributes in a Bitcoin transaction created by the provider. This transaction must be shared with the user along with a plain text signed version of the attributes. This will be later needed to provide attributes to relying parties, and serve as recovery in case the mobile device is lost. The encrypted version of certifications can be stored in a ShoCard server, this server never learns the encryption key, enabling the user to share certifications with selected parties only.

The last phase, the validation phase, occurs when a relying party does a verification to determine whether a user is entitled to access a service. To validate the encrypted certification the user provides the relying party with a reference and a key to it. The relying party then proceeds to do all necessary verifications: the key that signed the encrypted certification is the same that signed the seal; the certification signature is from a trusted entity and the plain text certification corresponds to the one hashed in the certification; and that the identity presented by the user in the transaction match those signed and hashed in the seal.

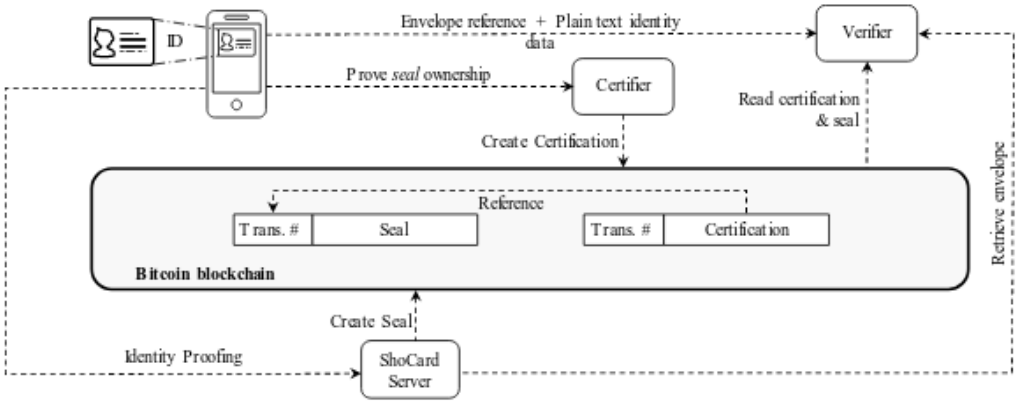

Figure 4.4: ShoCard uses Bitcoin to record personal data that was verified during identity proofing, and for storage of hashes of certifications that build upon the Seal of the user created by relying parties. The ShoCard server plays an active role as an intermediary between users and relying parties.

Chapter 5

Solution architecture

An entity that publishes attributes about a subject must have some level of trust. One can say that the value of that attribute will be as high as the trust in the Attribute Provider (AP) that published it. Frequently the trust deposited in an AP will not be flat, but hierarchical, i.e. behind an AP should be an entity that the other peers hold in higher regard. To address this, we intent to explore entities that are responsible for regulating certain sectors of business that usually are more trusted by the general public.

In this section we describe a system that aims to provide trust on personal attributes while trying to facilitate its management and disclosure by the owners.

5.1 Roles

Four types of roles were defined:

1. **User**: this role is to be taken by entities interested in having their attributes certified;
2. **Attribute Provider (AP)**: this role is to be taken by entities entitled to certify attributes belonging to a given subject (e.g., a university);
3. **Service Provider (SP)**: this role is to be taken by entities interested in consume attributes claimed by users and, for that reason, must verify its trustworthiness;
4. **Regulatory Body (RB)**: this role is to be taken by entities in charge of regulating or supervising activities in a particular business sector (e.g., ANACOM, the authority that regulates telecommunications and postal services in Portugal, etc.).

The RBs must also be a member of the consortium responsible manage the system. This should guarantee that the administrator will not be seen as a "higher than everything" entity.

5.2 Attributes

A certified attribute is a set of attributes with two or three extra attributes: an ID of the issuer (public key), the ID of the owner (public key), a digital signature made by the issuer and the personal attribute.

A personal attribute is a tagged value; the tag describes its semantics, e.g., birthday date, weight, address, etc.

```
{
  "id": "did:example:ebfeb1f712ebc6f1c276e12ec21",
  "type": ["Entity", "Person"],
  "name": "Alice Bobman",
  "email": "alice@example.com",
  "birthDate": "1985-12-14",
  "telephone": "12345678910"
}
```


Figure 5.1: Example of a set of attribute as a tagged value

An attribute seldom is used alone but rather grouped with other attributes. This happens because hardly a single attribute describes all the characteristics of an entity required in some context. Typically, at least two attributes are needed: one with an identifier (that identifies the entity to which the attributes refer) and the other containing the entity relevant characteristic for the given context. Figure 5.1 illustrates a set of attributes. Other examples of grouping of attributes are the user attributes in a driver license or in a national Id card (e.g., Portuguese Citizen Card).

Attributes can have a lifetime or not, this depends on the AP that issues them. Attributes can also have a infinite or finite number of disclosures, it also depends on the issuer. We could consider a document that needs to be renewed from time to time (e.g., a student identification card, which is renewed annually) as prime example of an attribute with lifetime, typically these certifications only have a validity of three years and need to be renewed after that time.

As an attribute is to be linked to an unique address, any and every update on this attribute will be readily available to anyone who as been give access to it. This means that if a user is to reveal their home address to an employer and other services, if they move to a different house they would change the attribute that corresponds to their new home address and the change would be propagated.

5.3 Interactions Overview

The interactions between the roles described must be governed by a basic set of policies. These policies describe the requirements that entities must fulfill in order to be able to participate in the network and how this participation is made.

5.3.1 Attribute certification

The client that approaches an AP to request the publishing of an attribute assumes the role of User. It is the responsibility of the AP to employ a process of authentication of the user at the moment of the request before making any certification. After this they should agree on the format of the attribute, and the user should provide details relative to the wallet address with which the attribute will be associated.

5.3.2 Exploitation of certified attributes

The wallet where the attribute is stored contains a set of secret credentials (i.e., a pair of private-public keys), that allows the user to assert the ownership of the attribute. Thus, when a User interacts with a SP, to assert possession of an attribute the user must provide a reference to said attribute and use the keys to sign it to prove ownership.

The AP must not be able to monitor the usage of the attribute, meaning that he can not limit or know to whom the attribute is presented.

At any time an attribute can be revoked by the AP, this operation does not eliminate the certificated attribute from the ledger, it only translates into a impossibility to use it as a valid attribute after that date. A revoked attribute can still be shared with a SP to state that we owned such attribute in the past.

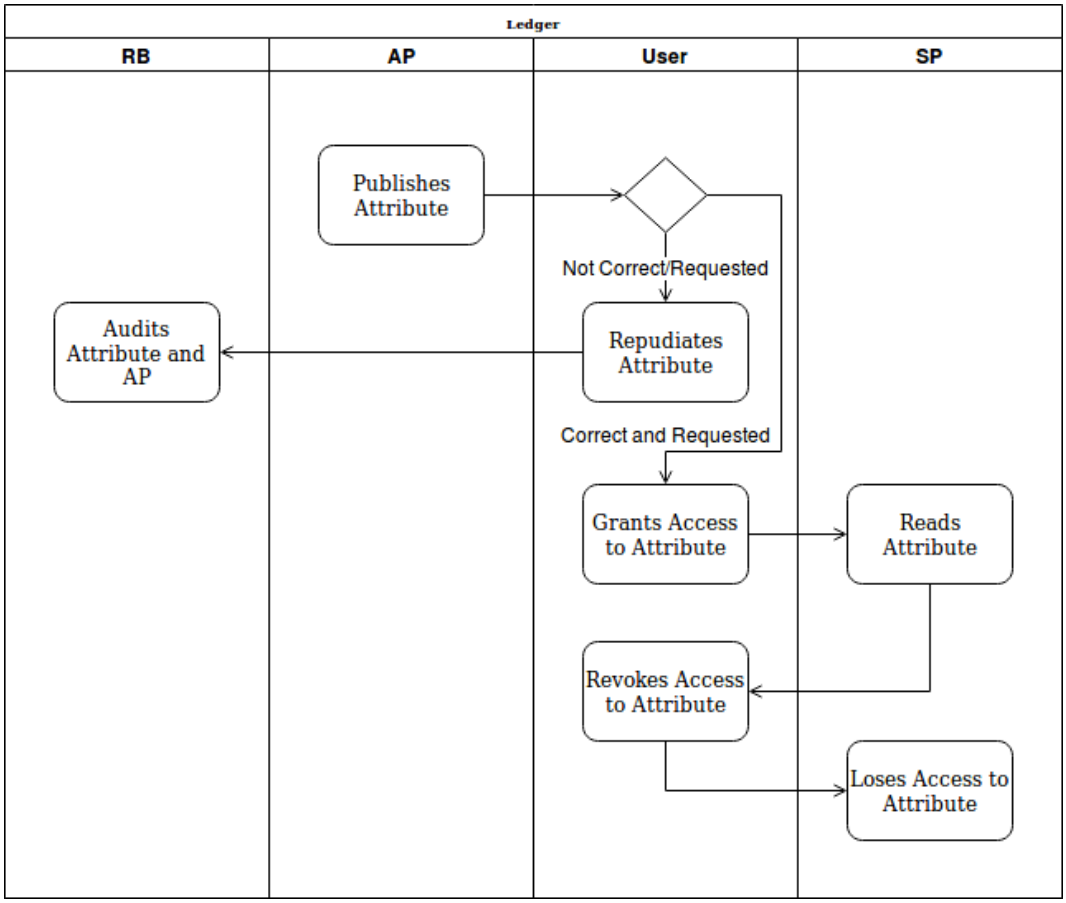

Figure 5.2: Flow of actions. The normal flow can be disrupted by a rogue attribute that would trigger a repudiation and consequently an audit. Not illustrated in the diagram are the operations of attribute revocation made by the AP.

5.3.3 Privacy

Each certificate is bound to a unique User pseudonym, the set of pseudonyms is managed by a user wallet. With this it is impossible to know which is the real identity of the owner of an attribute or set of attributes. Furthermore, attributes can either be obfuscated or in clear value.

5.4 Trust Management

The society, in general, trusts on RBs to do a proper supervision of their business sector and to keep a coherent blockchain among themselves. Within RBs, they only trust on authenticated RB peers to enter on blockchain definition update agreements.

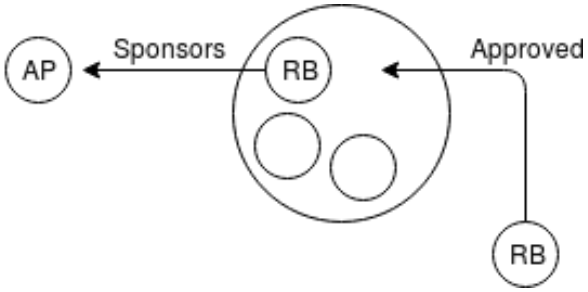

Figure 5.3: In the proposed trust model, every new RB needs to be approved by pre-existing RBs in the ledger. New APs are approved by their respective RB

Each RB maintains a list of federated AP and trusts that APs perform a correct certification. Abusive certifications cannot be immediately detected by RB but can be later identified by auditing transaction or by attribute repudiation by the User.

To use the attribute the SP only needs to trust the AP that issued the attribute and verified that the User is indeed the owner. This can be seen as a similarity to a PKI. However this similarity is limited to the hierarchy, as the objectives of each are different.

In a PKI, the goal is to offer trust in certified attributes (a certificate is nothing more than a set of certified attributes, and in that sense a CA may be seen as an AP), i.e., it guarantees that an entity that certifies attributes is who it say it is. With this in mind, a PKI should be employed so that the consumer of attributes knows who was the entity who certified such attribute.

This hierarchy aims to ensure that the APs that participate in the ledger are legitimate, and that they can issue certified attributes of some sort. It should be noticed that there may be some APs of more than one type. In essence, this hierarchy guarantees than the AP that issues certificates is entitled to do so, but it does not mean that it is authenticated.

5.5 User Wallet

A user wallet is fundamental to manage a set of attributes. It assures the secrecy of the credentials used by its owner, both when requesting and disclosing certified attributes.

The wallet is the main point of interaction between the user and the network and must be able to shield the user from technical details. This details may include, but not be limited to, deterministic hierarchical key management. The introduction of this standard would allow the user to have multiple attributes, each within a different address, while only needing to manage one master key.

The management of keys could also be simplified with the introduction of BIP-0039[1], this standard introduces a pass-phrase or mnemonic sentence to generate and manage key

[1]https://github.com/bitcoin/bips/blob/master/bip-0039.mediawiki

pairs, this type of element is more user friendly than handling raw binary or hexadecimal representations of a wallet seed.

Chapter 6

Implementation

6.1 Choice of Blockchain

As mentioned before, and summarised in table 3.1, there are three categories of blockchain.

Figure 6.1 shows a flow chart intended to help choosing between a traditional database, a permissioned blockchain or public blockchain. The questions presented help make the right decision based on the requirements defined.

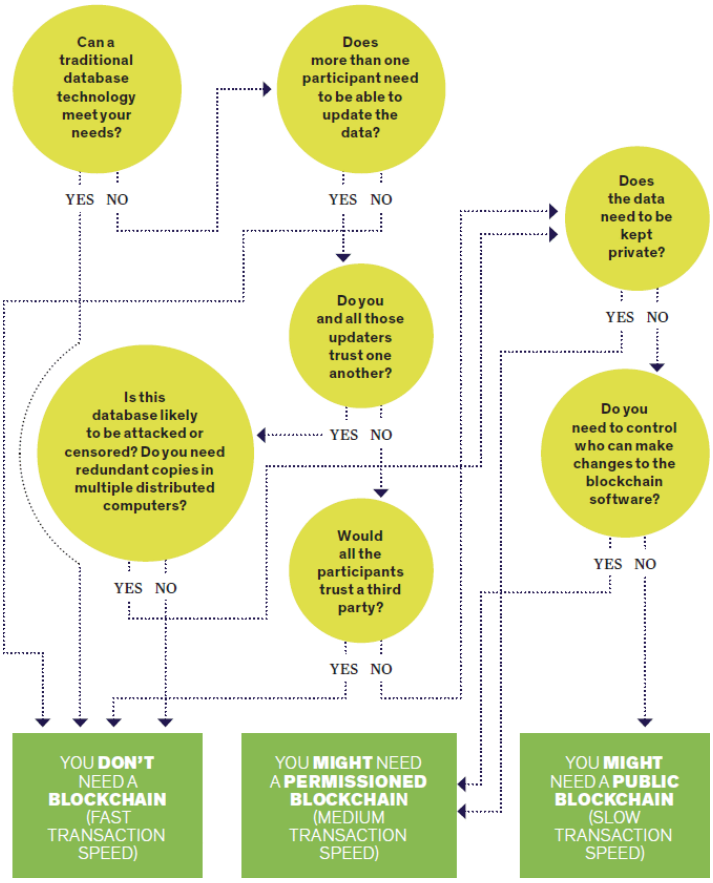

Figure 6.1: Graph to answer: Do you need a blockchain? [33]

Confronting this chart with our requirements we are able to justify the choice of a permissioned blockchain because of the following:

- The traditional database cannot meet the needs of self-sovereign identity;
- More than one entity need to be able to update the data;
- There is no trust between all participants;
- There is no third party with full trust;
- The data must be kept private;
- There is the need to control read and write access to data.

The necessity to employ a permissioned ledger is highlighted by the answers given to the questions presented by 6.1. There are three implementations of permissioned blockchains: Hyperleder[1] Fabric; Corda[2]; Multichain[3].

Comparing this three alternatives, Hyperledger is the strongest option based on the following:

- **Modularity**: The modularity of Hyperledger enables organizations to plug their preferred encryption, consensus and other components that may integrate well with existing systems.
- **Transaction Speed**: Transaction confirmation is rated at 100 000 transactions per second.
- **Safety**: Identity management and strong authentication provided by PKCS11.
- **Smart Contracts**: Permits more comprehensive capabilities.

6.2 Hyperledger Fabric and Hyperledger Composer

In this thesis, Hyperledger Fabric[4] and Hyperledger Composer[5] are used as blockchain framework and developer tools, with more focus on the later. Meaning that we can abstract from more advanced and tweakable features of Hyperledger Fabric that escape the scope of the project.

However, the necessity to understand the big picture that fabric provides is of great importance.

[1] https://hyperledger.github.io/
[2] https://www.corda.net/
[3] https://www.multichain.com/
[4] https://github.com/hyperledger/fabric
[5] https://github.com/hyperledger/composer

6.2.1 Architecture Overview

The architecture of a system built on top of Fabric and Composer is distributed in three layers: Application, Developer Tools and Blockchain Runtime.

Communication between the application layer and tool layer is made through a REST API, this API is provided by a rest server embed in the tools layer. The connection between the other layers, tools and runtime, are made with secure connection profiles. This profiles contain TCP/IP address and ports for the peers and certificates. Figure 6.2 tries to summarise interactions.

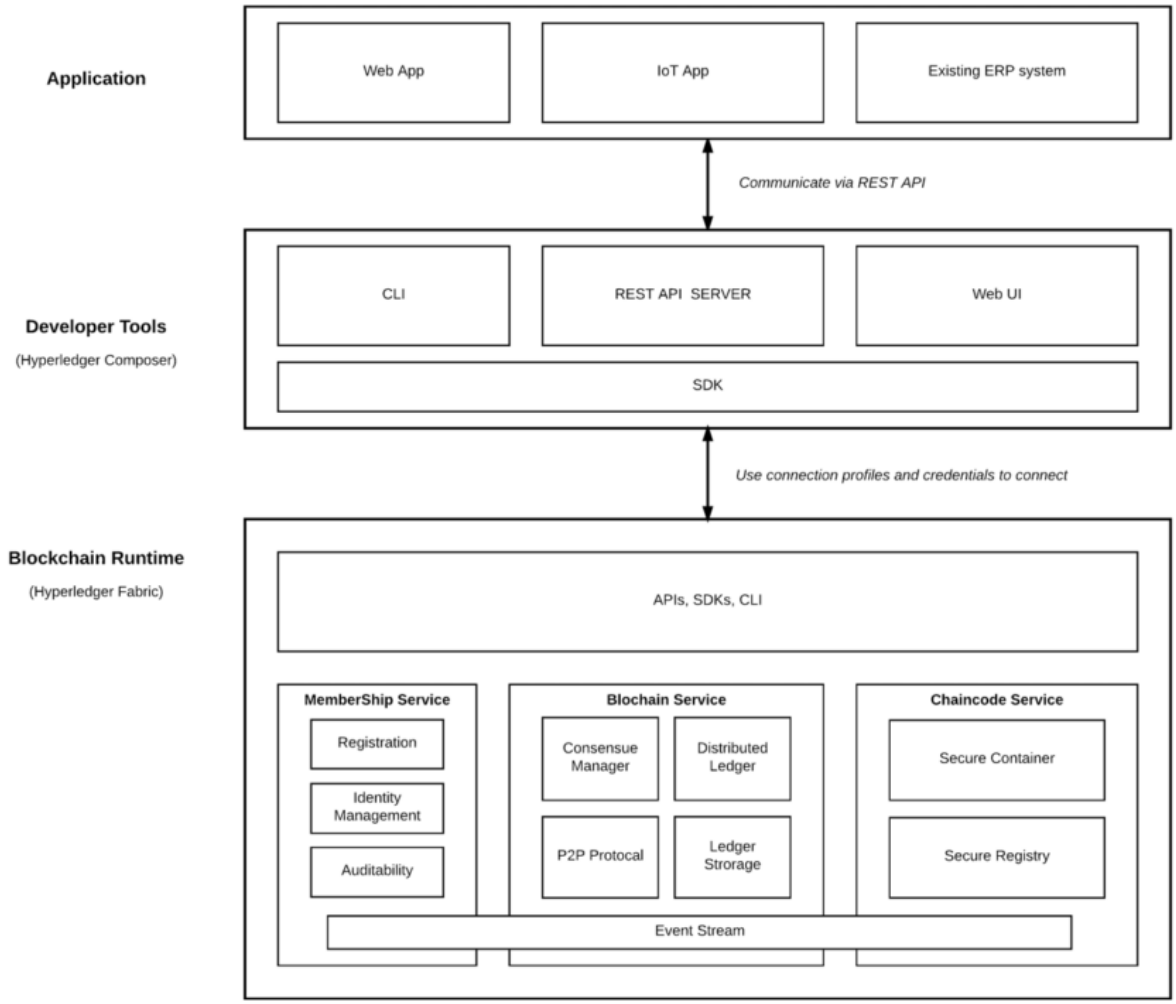

Figure 6.2: Hyperledger architecture

The application layer may include web applications, existing systems, or in some use cases Internet of Things (IoT) devices.

In the developer tools layer, Hyperledger composer provide a vast toolset to enable rapid development and testing of business logic intended to run on top of a blockchain. It offers some high-level components: Command Line Interface (CLI), Rest server for integration with applications, a Web UI, and a SDK. The Web UI is a playground that allows the developer

to rapidly engage with the network and test the network definition. The later is a API built on top of Node.JS that allow us to perform operations on blockchain resources.

In the Blockchain Runtime component, we have Hyperledger Fabric, the distributed ledger. In this layer we have: Membership, Blockchain and Chaincode (i.e., smart contracts) services.

6.2.2 Consensus Flow

In hyperledger the consensus protocol can be adapted to the necessities of the network, however it provides a default consensus through Kafka Ordering Service. The ledger reaches consensus by performing two activities: Ordering and Validating Transactions.

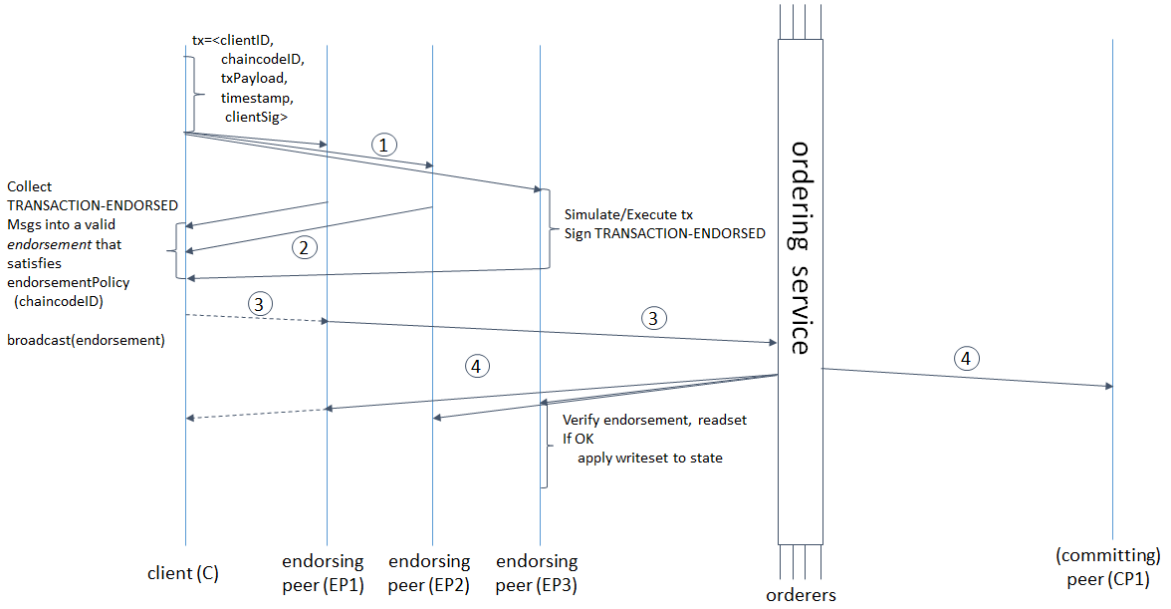

Figure 6.3: Hyperledger Consensus Flow[6].

The client submits transactions to a pending pool of consensus. The order will then select a specific number of transactions and order them based on the consensus algorithm, this can be, as mention, adapted. The contents of transactions are encrypted so that the orderer remains agnostic to the transactions.

The next step is validation by the chaincode, the goal is to check if the transactions are in compliance with the logic defined before. If the transaction is verified with success it will be broadcasted with proof of correctness. In the end the changes are committed to other peers in the ledger.

6.3 Modeling of network

In Hyperledger Composer there are four main types of resources: Assets, Participants, Transactions, and Events. These are succinctly explained in table 6.1.

[6]Image source: https://hyperledger-fabric.readthedocs.io/en/latest/arch-deep-dive.html

	Definition	Example
Asset	Tangible or intangible goods, services or property.	Houses and listings.
Participant	Actor with a certain role.	Buyers, owners and realtors.
Transaction	Interactions between participants and assets	Buying, selling, and creating and closing listings.
Event	Message emitted by transaction.	"Alice bought a house from Bob"

Table 6.1: Summary definition and examples of resources on Hyperledger

Other types of classes such as enumerates, types and concepts are also defined within the model of the network, so they all fall in the same category and can also be consider resources.

The specifics of assets, participants and transactions, and how this translates for this solution are detailed below.

6.3.1 Assets

Assets can range from tangible to intangible things, i.e it can be something that has a physical representation or something without a physical form. In our solution there is an intangible asset, the attribute.

The attribute must, inevitably, be linked to the owner and the issuer. The linkage to the owner is very important, as in self-sovereignty we need to guarantee that the owner is who ultimately controls the aspects about them. At request from the user they will have the power to update the attribute, and, in case of need, they can revoke it.

The act of revoking an attribute must be justified it can not be a simple binary property. The status of revocation will have to comprise both the binary status (i.e., valid or not valid) and a field with details about it.

In certain situations it also may be of use having other entities capable of revoking a certain attribute. Be it in the case of also having some responsibility about the emission of that attribute or in the case of a rogue AP that is being closed down by the respective RB.

This can be summarized as saying that the essential properties to define this asset are: an ID imposed by the modelling language, a field containing the actual attribute (i.e. a blob), an owner, an issuer, information regarding the attribute validity or revocation status, and a list of optional revokers. Besides this essential properties, the attribute will also have a field stating the type of business that issued it.

The owner will have the power to reveal their attributes to other participants in the network, having this ability means that the owner must always be aware of which entities are capable of reading it. This can be done by adding a field with list of authorized users. Unfortunately, in the current state of the Hyperledger Composer, the access control of asset properties is not supported. Adopting this approach would allow entities with access to the attribute to know all others that also have access, raising a linkability problem.

6.3.2 Participants

The participants are the actors with specific roles in the network. Previously we identified four distinct roles: RBs, APs, Users and SP.

All of them have the need to have an ID and digital certificate containing their well known public key. To this generic type of participant we are going to call it "Member", and upon it the other participants will be built.

For the RB we will add a field with the name of the institution (e.g., FCC) and the business sector they are responsible for. The AP is very similar, but instead of being responsible for a sector they belong to it.

The SP will have only one addition, a name, the goal of this field is to allow the user to better understand who are the entities that he is going to give permissions. In the definition of the SP it is also included a list of attributes to which they where granted access. This will prevent the linkability problem referred in the attribute description.

The User is the only participant with no extra fields compared with the Member.

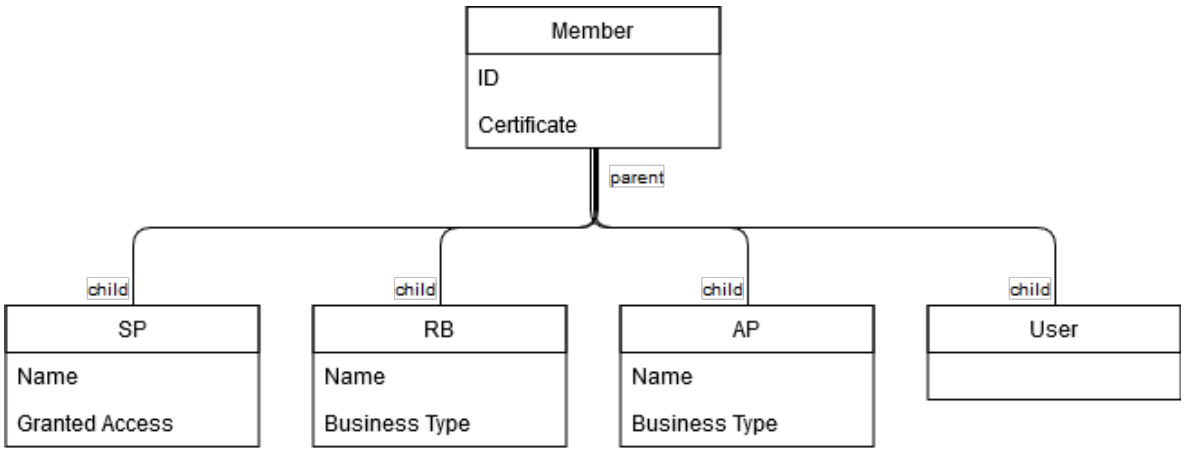

Figure 6.4: Participants properties

6.3.3 Transactions

Transactions are a mechanism of interaction between participants and other resources. A Participant can submit a transaction in order to create, delete or change one or more properties of assets. Transaction are particularly useful when there is the need to limit or complement default actions.

There are three interactions core to our proposal that cannot be translated to pre-existing transactions:

1. **Revoke/Repudiate attribute**: Even though a revoke operation can only be triggered by a AP and a repudiate operation can only be triggered by the owner of the attribute, the logic behind them is the same. A entity with permissions (i.e., a revoker) will submit this transaction where three mandatory fields need to be filled, attribute ID, revocation status and revocation details. If correct this details will correspond to a valid transaction and from that point on that attribute cannot be used again. This does not mean that it is deleted, it will remain in the user wallet, and can serve as proof of past validity.

2. **Grant Access**: When Users wants to allow a SP to read and validate one of their attributes, they will need to submit a grant access transaction from their wallet. Ap-

plication can provide a certain level of abstraction from this operation and offer a more user friendly approach, but this transaction receives an attribute ID and a SP ID. When successful, this transaction will append the attribute reference to the aforementioned SP authorized read list.

3. **Revoke Access**: Working very similarly to the previously described interaction, this transaction instead of appending the attribute reference will remove it.

Figure 6.5 illustrate a transaction processor function, more specifically the one corresponding to grant access operation.

```
/*
 * Track granted authorizations to access attributes
 * @param {pt.ua.attr.GrantAccess} transaction - grant access
 * @transaction
 */
 async function GrantAccess(tx){
    //required registries for this operation
    const participantReg = await getParticipantRegistry(namespace + '.SP');
    const assetReg = await getAssetRegistry(namespace + '.Attribute');

    let sp = await participantReg.get(tx.memberID);
    let attr = await assetReg.get(tx.attrID);
    let attrID = tx.attrID;
    let index = -1;

    if(!sp.granted){
        sp.granted = [];
    }
    else {
        index = sp.granted.indexOf(attrID);
    }

    if(index < 0){
        sp.granted.push(attrID);
        await participantReg.update(sp);
    }
}
```

Figure 6.5: Example of a transaction processor function

6.4 Access Control

Access Control Language (ACL) allow us to define access control rules of interaction between asset, participant and transactions. With it we can restrict which participant types are permitted to operate resources in the network.

Is important to refer that there are two types of access control: business access control and network control. They control the access to the resources within the network and the administrative changes, respectively.

Rules can be simple or conditional. Simple rules are used to control access to an asset, or property, of an asset by a participant. On the other hand conditional rules employ a Boolean JavaScript expression in order to evaluate the application. In the later we can indicate a transaction, and this will allow the participant to access the resource only through that transaction.

```
rule executeGrantAccessTransaction{
  description: "Allow participants to submit GrantAccess transactions"
  participant: "pt.ua.attr.Member"
  operation: CREATE
  resource: "pt.ua.attr.GrantAccess"
  action: ALLOW
}

rule grantAccessTransaction{
  description: "Allow participants to submit GrantAccess transactions"
  participant: "pt.ua.attr.Member"
  operation: UPDATE, READ
  resource: "pt.ua.attr.SP"
  transaction: "pt.ua.attr.GrantAccess"
  action: ALLOW
}
```

Figure 6.6: Example of Access Control Rules

The figure above shows an example of network access control. This case shows us how a transaction can limit the access to resources. In a normal scenario it is inconceivable that a participant can alter properties of other participants, but in this case, through the transaction, the user can append the authorization in the SP.

6.5 Integration Interface

The business network can be integrated with exiting systems or other applications using a Loopback[7] API. This provides a feasible way to have systems sending data to the blockchain.

Hyperledger Composer exposes a REST API that serves the purpose of a gateway, subscribing events triggered by transactions. The server and client communicate through a WebSocket.

The REST server can be secured with HTTPS and TLS, allowing encryption of all data between client and server.

The transaction processor can also call external REST services in order to move complex computation off the blockchain.

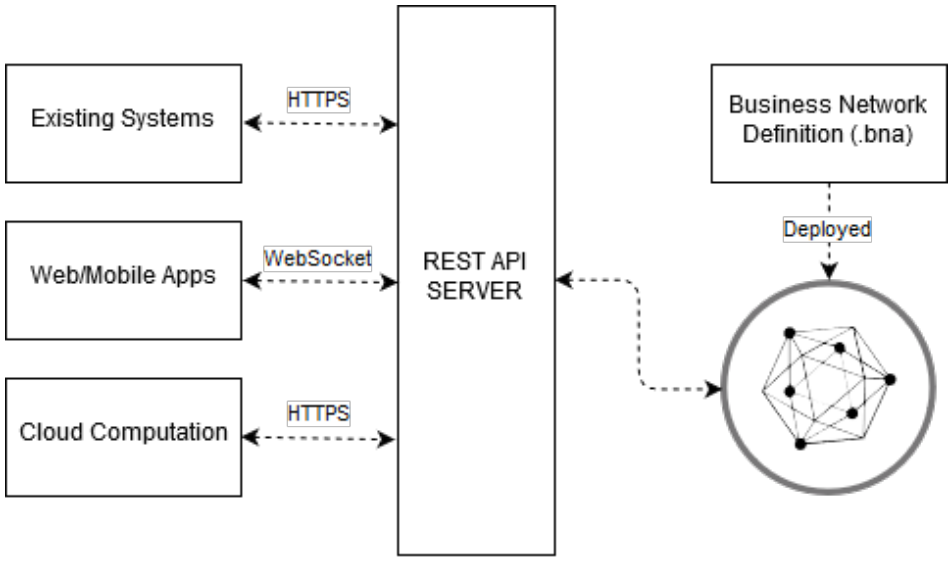

Figure 6.7: Integration with other services

This offers the possibility to securely integrate the network with external services with different protocols. Thus a simple web app was devised to test said integration capabilities and other aspects of the system.

6.6 Testing and Demo

6.6.1 Testing

In order to test iterations of the network definition we employed a bash scrip. This script purges the network, creates the archive, installs the definition on the containers, authenticates the administrator and launches the network.

After initializing the containers and bring the network on-line, we were able to explore the REST API and use another script as genesis of the network, creating resources so that particularly interactions could be tested with the assurance that the resources were well defined and compliant.

The composer playground offers a easy to use interface to test the intended features of a network definition.

[7] https://loopback.io/

Figure 6.8: Composer playground: entities used in testing

Figure 6.8 shows us a list of all the entities used during tests. Composer also allowed us to keep all the entities in the same wallet so we could swap between one another.

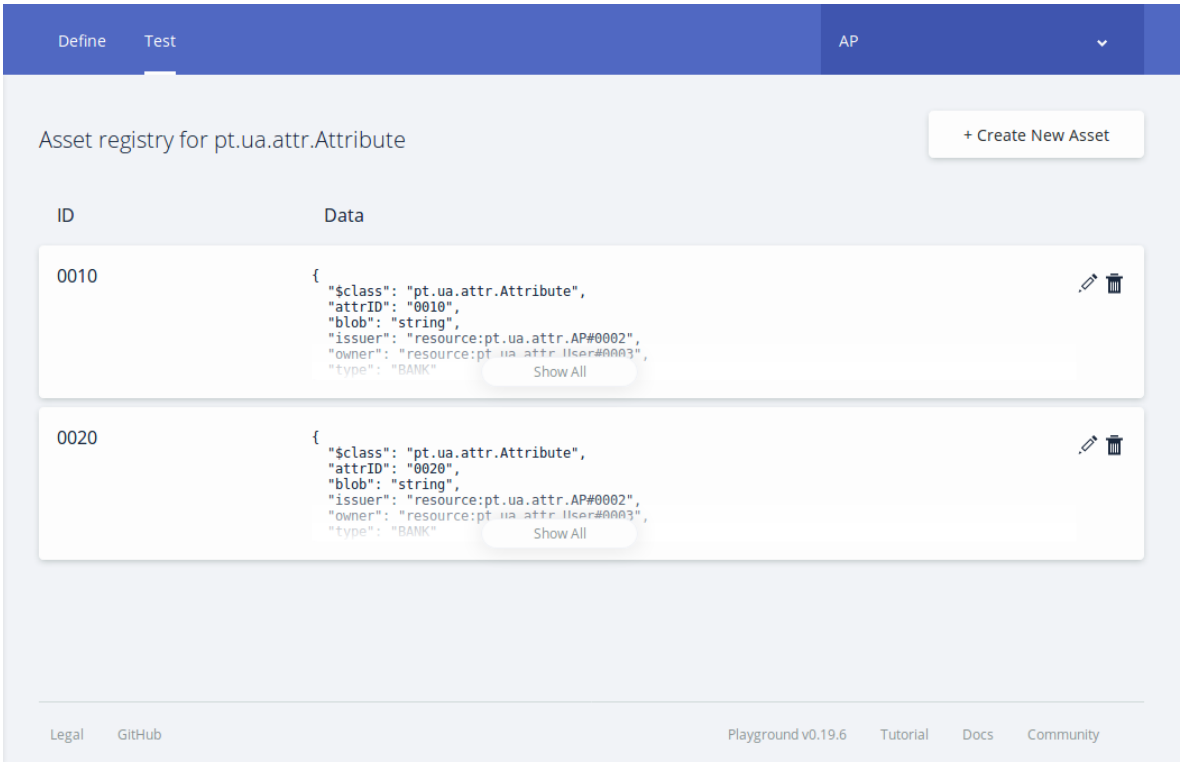

Figure 6.9: Composer playground: Attribute Provider view

Figure 6.9 represents the view of the asset registry while logged as the AP with the ID 0002. As intended the AP has read permissions over the attributes issued by them. The same registry would contain different results depending on the entity that was logged.

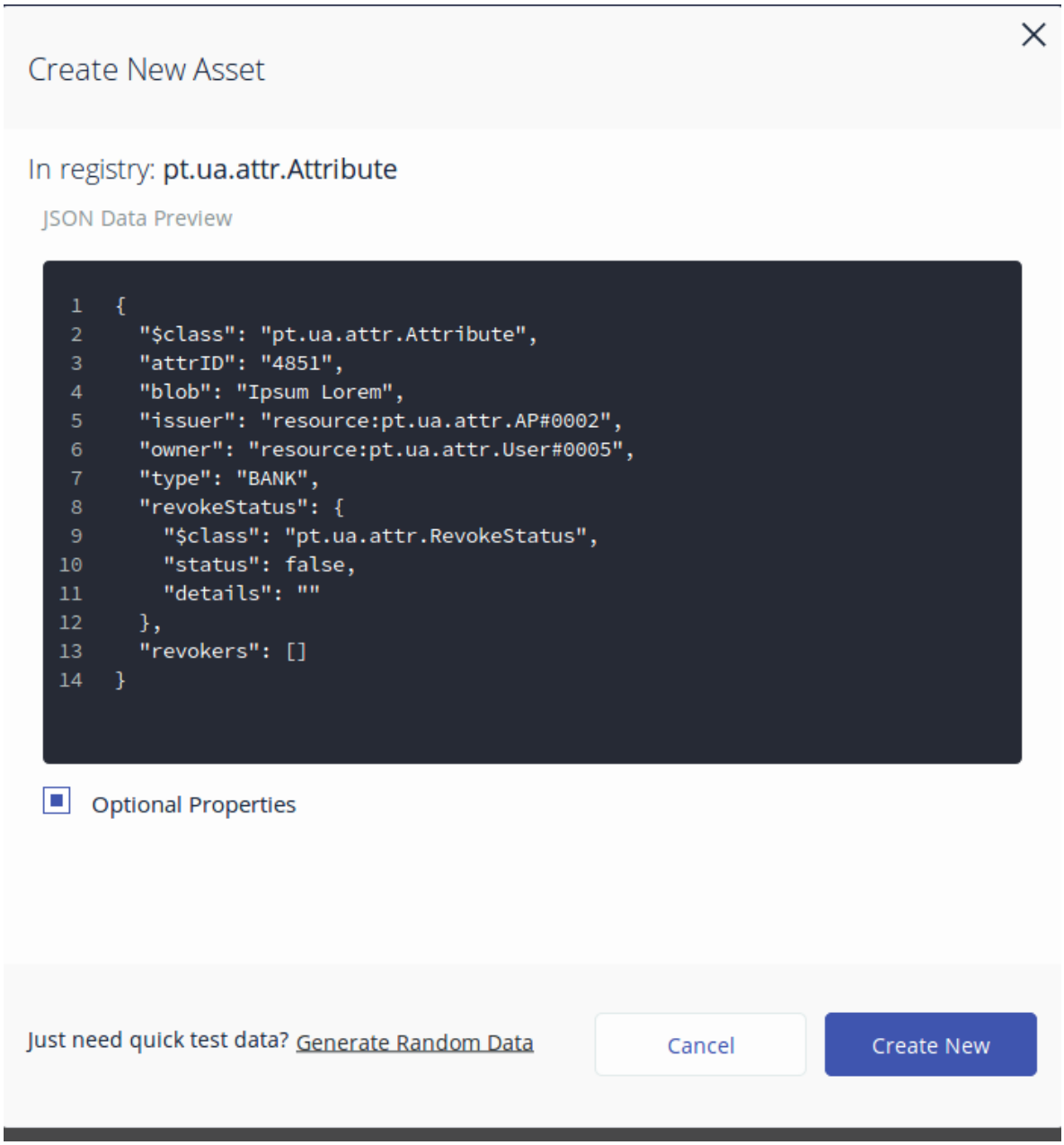

Figure 6.10: Composer playground: Asset Creation

Figure 6.10 contains the form for asset creation. Asset creation, as well as participant creation, are pre-existing transactions. Being pre-existing transactions they have dedicated buttons to activate them. Once again this allows us to test several conditions intended to be controlled by the access control module. In this case the ledger would verify that the issuer number in the JSON is the same of the entity logged in, as well as the business sector.

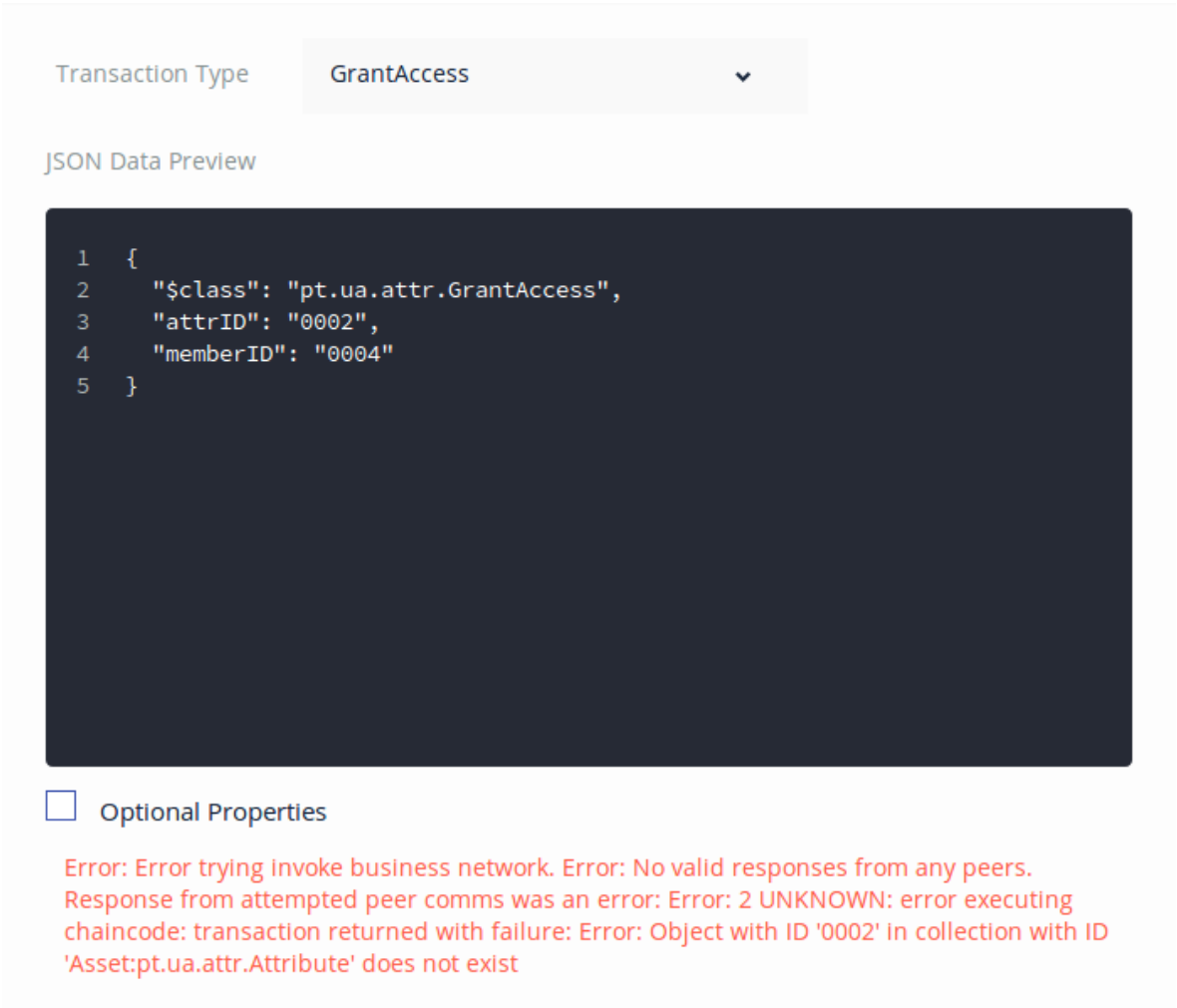

Figure 6.11: User App Grant Access action

Figure 6.11 is an example of a failed transaction. For testing purposes we have an AP submitting a grant access transaction for an asset that belongs to one of their clients. Given that he does not own said asset, the transaction naturally fails, the lack of read permissions over the object triggers an error when the nodes try to execute the chain code during the endorsing process.

The results of testing will be discussed in detail in chapter 7.

6.6.2 Demo

A simple demo was made to complement the testing capabilities offered by the Hyperledger Composer playground. The later serves its propose to test a prototype network but does not allow us to abstract from some details, such as fields that are automatically defined. It was also built with the intent to test the integration through the REST API.

Instead of a raw JSON, the demo has a friendlier approach to transactions. In it, the user is prompted with a form where some fields are already filled, needing only to indicate the

SP ID to execute a grant access transaction, the behaviour will be exactly the same in the opposite operation.

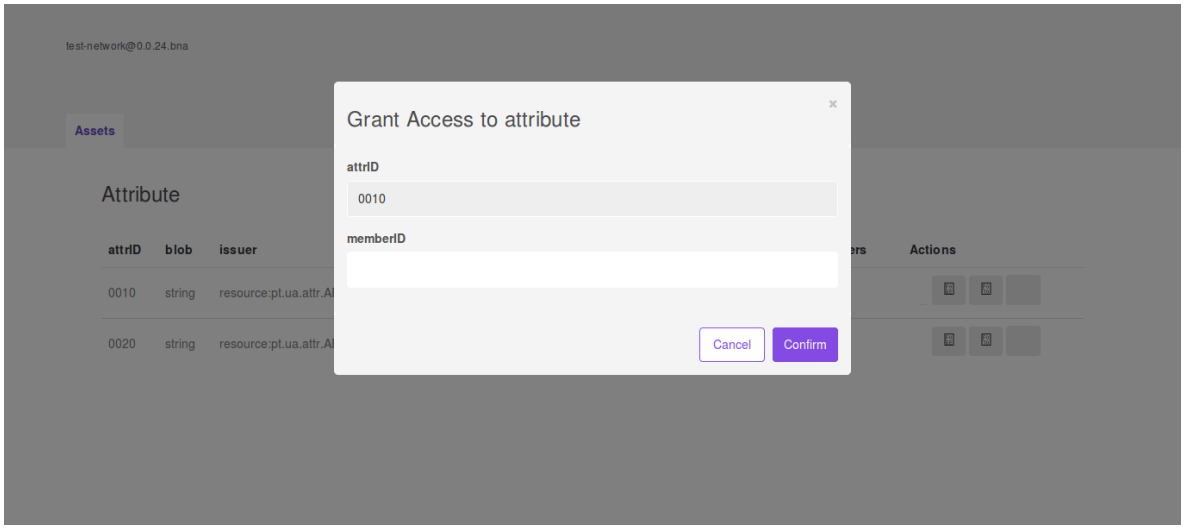

Figure 6.12: [Composer playground: Failed transaction

The Demo also allow us to get a different view on some interactions. The most notorious example of this is when a SP as granted access. Without the app we would have to check the SP listings as a user to find the SP with access to our attributes.

Using the composer playground, we would add entities with the various roles to our wallet and alternate between them to test read and write accesses.

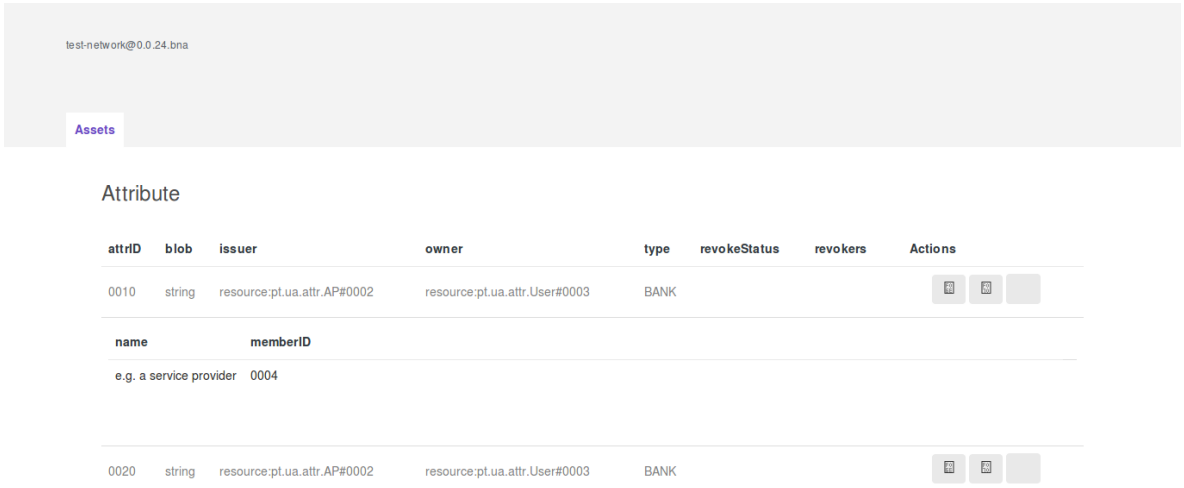

Figure 6.13: User App listing SPs with access to an attribute

Chapter 7

Result Analysis

As stated in chapter 1, the goal of this dissertation is to devise a decentralized network capable of supporting validation of personal attributes and enforcing the principals that define a self-sovereign identity as well as data-security, issuer validation, unlikability, revocation and usability.

Being employed on a permissioned ledger means that the network can force a certain level of regulation through code, if done correctly, this code will always operate as intended and maintain policies across the network no matter what. In Hyperledger case, this is in great part done by the access control rules as described in section 6.4.

After some analysis on all interfaces made available to interact with the network, the REST API, the Composer Playground, and the simple user demo, we can make some judgement on how said rules enforce our goals. In a situation where a participant does not have permission to READ a object the system will respond with an error claiming that the object does not exist. This is interesting in a sense that an evil doer cannot test the existence of random attribute IDs. Rules promote data security, user access and control over their attribute.


```
{
  "error": {
    "statusCode": 404,
    "name": "Error",
    "message": "Unknown \"Attribute\" id \"0030\".",
    "status": 404,
    "code": "MODEL_NOT_FOUND",
    "stack": "Error: Unknown \"Attribute\" id \"0030\".\n    at Function.convertNullToNotFoundError
  }
}
```


Figure 7.1: Error upon trying to access an attribute that does not belong to the current user.

There is of course the need to authenticate participants through the Membership Service Provider (MSP), so the rules can know who the participant is supposed to be and which assets can he access, and how. This represents a contradiction to the principle of existence in self-sovereign identity, given that the user cannot be independent from the system, but instead is part of it. Even though it can be abstracted by saving the credentials next to the participant app, figure 7.2 shows the files that include details on how to access the network (i.e., address

and ports) and the cryptographic keys to authenticate the user.

Figure 7.2: Card files to authenticate participants

Using a decentralized system with centralized control means that a few principles of self-sovereign identity can only be achieved through a responsible consortium, this entity that is mapped into the administrator role must allow the other participants access to the definition of the network in order to cultivate transparency as stated in 2.1.2. We found that the use of a persistent ledger contradicts the right to be forgotten, even though the attributes cease to exist in the current state, they will forever be in the transaction history, even though only the consortium can access and audit it, this represent not only a shortcoming from our solution but also a shortcoming in the usage of blockchain technologies in self-sovereign identity.

Unlikability in a technology with the natural ability to keep a registry of all operation that occur represents a challenge, as stated in 6.3.1, poor planning can lead to the possibility of linking attributes to IDs and to providers. Even though the blob of the attribute may be obfuscated, and all the users hide behind an alias, one could infer many information by looking at the issuer of an attribute and to the services that are consulting it. Limiting the READ permissions on the transaction history is imperative, as well as storing access permissions separated from the attribute.

The major shortcoming in the current state is minimization, there is no possibility to reveal only certain properties of attributes or some form of simplification on this attributes, such as an assertion of being older than a certain age. Future work would need to be employed to achieve.

The enforcement of issuer validation is different, meaning that the transaction enforces the type of business of the attribute so that it is the same of the issuer at the moment of creation of the asset.

Usability is important as the user should be shielded from technical details, for the users it is not relevant if they are using a private key to sign a challenge in order to authenticate them, only that they need to use PIN to authenticate. This is also true in the demo built, there we can see two different approach to other entities evolved in the attribute. In one case, the issuer, we have a reference for is ID in the network, and in the other, we have the name of the service provider. This serves to compare which is more friendly, even though it could be obvious, it is important to support these conclusions with experimentation.

Chapter 8

Conclusion

This dissertation aimed to build a decentralized solution for personal attribute management that would follow the paradigm of self-sovereign identity. This was made using Hyperledger Fabric and Hyperledger Composer, as our research revealed that we had to employ a permissioned ledger and this was the one that better suited the needs raised by our goals.

Following an analysis of existing similar solutions, to bring self-sovereignty to reality, we concluded that true decentralization seems difficult and those solutions tend to leverage several decentralization techniques but end up reshaping the role of central entities. This unwanted centralization originates in the need to use central authorities as trust providers. During this research it was clear that this is an aspect that cannot be escaped because in context of identity we need to ensure trust. While DLT were designed to eliminate central points of control, this is not a realistic goal in scenarios that need to resort to IdM given the profound need for a source of trust.

Our solution shares the same flaws while dealing with attributes. We differ from the other proposals in the sense that the user does not make claims and then asks for their verification, instead it is an AP that directly creates and attests the claim. There is also an attempt to increase the trust by having RBs being responsible for approving issuers, and monitoring their actions, increasing the regulatory properties of the ledger. In the end, the user will still maintain the need to login into the new service provided by the network that we devised, creating one more credential to manage.

It is also important to refer that the tightening of regulations, of which GDPR is an example, gives end-users new powers over their data, and data controllers other responsibilities. This regulations also provide more transparency to the processes, and protection to the user rights. Unfortunately, decentralized solutions associated with immutable ledger, currently, are incapable of fulfilling some of the requirements imposed by those regulations (e.g., right to be forgotten). However, its positive to see legal initiatives protecting the user and adhering to some of the principles behind self-sovereign identity.

Delaying advances on this field of research may seem unacceptable as decentralization gains ground as the next best thing. We already see the original cartoon of Peter Steiner being replaced by the on-line adage "On the blockchain nobody knows you're a fridge"[1].

Working on this project, we have, not only, learned a lot about blockchain, but also about the way that technology can shape socio-economic practices around the concept of credentials.

Many of the most interesting challenges we encountered were not technical in nature,

[1] https://gendal.me/2013/10/23/on-the-blockchain-nobody-knows-youre-a-fridge/

but they cannot easily be separated from the technology because small design decisions can fundamentally shape behaviour. Blockchain is a maturing technology and its complexity and immutability make it even more important to consider the long-term effects of architecture decisions.

8.1 Future Work

The current network definition does not allow minimalization principle of self-sovereignty as intended, to support it would require the reimplementation of the current solution. Which means that the network definition as it stands is subject to changes, minimization would, possibly, require the introduction of new assets. This small aspect of self-sovereign should not be underestimated, an effective minimalization will protect individuals' privacy. This aspect of a self-sovereign identity system must be prioritized in future iterations of this system.

A permissioned blockchain would always be the best option, given that access control rules have a major role in securing the data on the ledger. Unfortunately it also means that the user must be, in certain way, enrolled in the system. It would be particularity interesting to be able to separate the user from the other entities, once again contributing for a solution closer from what self-sovereign identity means.

Since the scope of this thesis was mostly the definition of a regulated ledger capable of holding and distribute personal attributes, the definition of formats and obfuscation processes were not addressed. This must not undermine their importance and in future iterations it should be considered of high importance. Given that a big part of the logic behind obfuscation should be made before the attribute reaches the blockchain, applications directed at APs and SPs should be developed.

Bibliography

[1] Peter Steiner. On the internet nobody knows you are a dog. The New Yorker, July 5, 1993.

[2] Clare Sullivan. Digital Identity: An Emergent Legal Concept. University of Adelaide Press, 2011. https://www.adelaide.edu.au/press/titles/digital-identity/Digital_Identity_Ebook.pdf.

[3] ISO/IEC. Information technology – security techniques – a framework for identity management – part 1: Terminology and concepts. ISO/IEC 24760-1:2011, 2011.

[4] Gergely Alpár and Bart Jacobs. Credential Design in Attribute-Based Identity Management. TILTing Perspectives, 2013.

[5] Gabi Siboni and David Siman-Tov. Cyberspace Extortion: North Korea versus the United States. INSS Insight, (646), December 2014.

[6] Melanie Swan. Blockchain: Blueprint for a new economy. O'Reilly Media, February 2015.

[7] Pedro Coelho, André Zúquete, and Hélder Gomes. A propose for a federated ledger for regulated self-sovereignty. In Proc. of the 13th Iberian Conference on Information Systems and Technologies (CISTI), pages 1–4, Caceres, Spain, June 2018.

[8] Paul A. Grassi, Michael E. Garcia, and James L. Fenton. Digital identity guidelines. NIST Special Publication 800-63-3, June 2017. https://www.nist.gov/publications/digital-identity-guidelines.

[9] Steve Riley. It's Me, and Here's My Proof: Why Identity and Authentication Must Remain Distinct. Microsoft TechNet, February 2006. https://technet.microsoft.com/en-us/library/cc512578.aspx.

[10] RFC 2196 - Site Security Handbook, 1997. https://tools.ietf.org/pdf/rfc2196.pdf.

[11] Paul A. Grassi, Ellen M. Nadeau, Justin P. Richer, Sarah K. Squire, James L. Fenton, Naomi B. Lefkovitz, Jamie M. Danker, Yee-Yin Choong, and Kristen K. Greene. Digital identity Guidelines: Federation and Assertions. NIST Special Publication 800-63C, June 2017. https://www.nist.gov/publications/digital-identity-guidelines-federation-and-assertions.

[12] RFC 5849 - The OAuth 1.0 Protocol, 2010. http://www.rfc-editor.org/info/rfc5849.

[13] D. Hardt. RFC 6749 - The OAuth 2.0 Authorization Framework, 2012. http://www.rfc-editor.org/info/rfc6749.

[14] Whitfield Diffie and Martin E. Hellman. New Directions in Cryptography. IEEE Transactions on Information Theory, 22(6):644–654, 1976.

[15] R. L. Rivest, A. Shamir, and L. Adleman. A method for obtaining digital signatures and public-key cryptosystems. Communications of the ACM, 21(2):120–126, 1978.

[16] RSA Data Security. Understanding Public Key Infrastructure (PKI). An RSA Data Security White Paper, 1999. ftp://ftp.rsa.com/pub/pdfs/understanding_pki.pdf.

[17] S Turner. RFC 5755 - An Internet Attribute Certificate Profile for Authorization, 2010. http://www.rfc-editor.org/info/rfc5755.

[18] Shannon Appelcline, Dave Crocker, Randall Farmer, and Justin Newton. Rebranding the Web of Trust. White Paper from Rebooting the Web of Trust by, December 2015. https://www.weboftrust.info/downloads/rebranding-web-of-trust.pdf.

[19] Satoshi Nakamoto. Bitcoin: A Peer-to-Peer Electronic Cash System, 2008. https://bitcoin.org/bitcoin.pdf.

[20] Jaap-Henk Hoepman. Distributed Double Spending Prevention. In Bruce Christianson, Bruno Crispo, James A. Malcolm, and Michael Roe, editors, Security Protocols, volume 5964 LNCS, pages 152–165, Berlin, Heidelberg, 2010. Springer Berlin Heidelberg.

[21] Janno Siim. Proof-of-Stake. Research Seminar in Cryptography, 2017. https://courses.cs.ut.ee/MTAT.07.022/2017_fall/uploads/Main/janno-report-f17.pdf.

[22] Daniel Larimer. Delegated Proof-of-Stake (DPOS). White Paper, April 2014. https://bitcointalk.org/index.php?topic=558316.msg6082114#msg6082114.

[23] Stefano De Angelis, Leonardo Aniello, Roberto Baldoni, Federico Lombardi, Andrea Margheri, and Vladimiro Sassone. PBFT vs proof-of-authority: Applying the CAP theorem to permissioned blockchain. In Italian Conference on Cyber Security, January 2017.

[24] Leslie Lamport, Robert Shostak, and Marshall Pease. The Byzantine Generals Problem. ACM Transactions on Programming Languages and Systems, 4(3):382–401, 1982.

[25] Miguel Castro and Barbara Liskov. Practical Byzantine Fault Tolerance. In Proc. of the USENIX Symposium on Operating Systems Design and Implementation (OSDI '99), New Orleans, LA, USA, February 1999.

[26] David Mazières. The Stellar Consensus Protocol: A Federated Model for Internet-level Consensus, February 2016. https://www.stellar.org/papers/stellar-consensus-protocol.pdf.

[27] Nick Szabo. Smart Contracts: Formalizing and Securing Relationships on Public Networks. First Monday, 2(9), 1997.

[28] Kim Cameron. The Laws of Identity. Microsoft Corp, May 2005. https://msdn.microsoft.com/en-us/library/ms996456.aspx.

[29] Andrew Tobin and Drummond Reed. The Inevitable Rise of Self-Sovereign Identity. A white paper from the Sovrin Foundation, September 2016. https://sovrin.org/wp-content/uploads/2017/06/The-Inevitable-Rise-of-Self-Sovereign-Identity.pdf.

[30] Christian Lundkvist, Rouven Heck, Joel Torstensson, Zac Mitton, and Michael Sena. Uport: a Platform for Self-Sovereign Identity. Draft version, October 2016. http://blockchainlab.com/pdf/uPort_whitepaper_DRAFT20161020.pdf.

[31] Sita and ShoCard. Travel Identity of The Future. White Paper, 2016. https://shocard.com/wp-content/uploads/2016/11/travel-identity-of-the-future.pdf.

[32] ShoCard. Identity Management Verified Using the Blockchain. ShoCard with ShoCoin Tokens Whitepaper, February 2018. https://shocoin.io/wp-content/uploads/2018/03/ShoCoin-ICO-Whitepaper-Feb9-lower.pdf.

[33] M. E. Peck. Blockchain world - Do you need a blockchain? This chart will tell you if the technology can solve your problem. IEEE Spectrum, 54(10):38–60, October 2017.